

A 2 M

6, ave du Général de Gaulle
Hall A
78150 LE CHESNAY
☎ 954.91.13

Microcomputer Components



Data Book

February 1980





Copyright 1979, 1980 by Zilog, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Zilog.

Zilog assumes no responsibility for the use of any circuitry other than circuitry embodied in a Zilog product. No other circuit patent licenses are implied.

Microcomputer Components Data Book

Innovation and Integration

At Zilog two words define our competitive edge in the marketplace: innovation and integration.

At Zilog "integration" means more than designing an ever-greater number of functions into a chip of silicon. Instead, Zilog focuses on the integration of technologies. LSI design technologies are enhanced by experience in computer-based system

technologies, such as system architecture and system design.

Zilog offers microcomputers to the user in every form: from components and development systems to OEM board-level products and general-purpose microcomputer systems.

This data book describes Zilog's microprocessor components, the basic foundation for other microcomputer products. Zilog's

components product line includes the midrange, 8-bit Z80[®] microprocessor and its family of peripherals, the low-end 8-bit Z8[™] single-chip microcomputer, and the high-end 16-bit Z8000[™] microprocessor with its family of peripherals.

Table of Contents

Z-80 Family	1-1	1
Z-80 CPU.....	1-5	
Z-80 PIO.....	1-7	
Z-80 CTC.....	1-9	
Z-80 SIO.....	1-21	
Z-80 SIO/9.....	1-37	
Z-80 DART.....	1-39	
Z-80 DMA.....	1-51	
Z-UPC/U.....	1-69	
Z-CIO/U.....	1-71	
Z-FIO.....	1-73	
Z8000 Family	2-1	2
Z8001/Z8002 CPU.....	2-5	
Advanced Architectural Features of the Z8000 CPU.....	2-29	
The Zilog Z-Bus.....	2-43	
Z8010 MMU.....	2-59	
An Introduction to the Z8010 MMU.....	2-75	
Z8 MCU.....	2-95	
Z8034 UPC.....	2-113	
Z8030 SCC.....	2-115	
Z8036 CIO.....	2-117	
Z8038 FIO.....	2-119	
Z8060 FIFO.....	2-121	
Memories	3-1	3
Z6115 8K x 1 Dynamic RAM.....	3-3	
Z6116 16K x 1 Dynamic RAM.....	3-13	
Z6132 4K x 8 Quasi-Static RAM.....	3-23	
Package Dimensions	4-1	4
Sales Offices	5-1	5

1

Z-80 Family
Zilog

Zilog Z-80® Family



New Peripheral Controllers Fit General-purpose Bus

1

February 1980

Three new peripheral controllers give fresh life to existing microprocessor applications while greatly broadening the application horizon of the Zilog Family of microprocessor components. These peripherals are evidence of Zilog's commitment to ongoing enhancement and support of the Z-80 Family.

The design philosophy of all Z-80 Family members is to help engineers design microcomputer systems with fewer components that have more functions per chip. The Z-80 CPU offers many more features and functions than its main competitor. All Zilog peripheral controllers are specialized

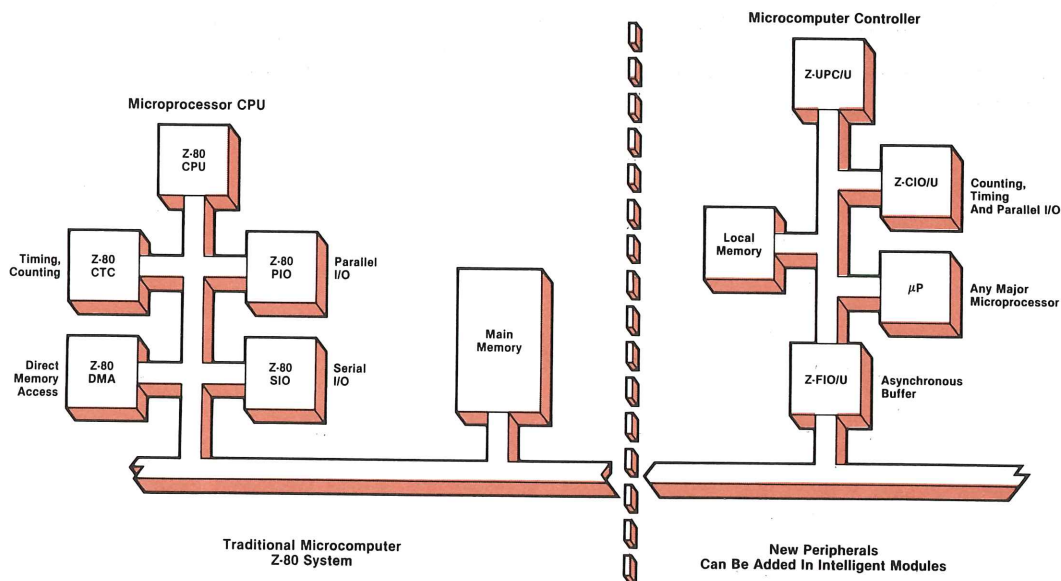
processors dedicated to providing functions that previously required a much larger amount of less-integrated circuitry or a great deal of software overhead.

Each extensively programmable peripheral contributes to unburdening the CPU and increasing system throughput. All Z-80 Family members and Universal Bus Peripherals can operate in interrupt-driven or polled environments. A daisy-chained interrupt structure prioritizes peripheral interrupts to the CPU, saving an external interrupt controller.

Chip count is further reduced both by multiple channels and extensively programmable

capabilities. The CPU addresses peripherals using a common I/O command structure that tailors each peripheral to the precise configuration required by the application.

The **Z-80 CPU Central Processing Unit** has rapidly established itself as the most sophisticated, most powerful and most versatile 8-bit microprocessor. Although source-code compatible with the 8080A microprocessor, the Z-80 CPU offers many more instructions (158 vs 78) and numerous other features that simplify hardware requirements and reduce programming effort while increasing throughput.



The dual-register set of the Z-80 CPU allows high-speed context switching and more efficient interrupt processing. Two index registers give additional memory addressing flexibility and simplify the programming task. Interfacing to dynamic memory is simplified by on-chip programmable refresh logic. Block move plus string- and bit-manipulation instructions reduce programming effort, program size and execution time.

The Traditional System. The four traditional microcomputer-system functions—parallel I/O, serial I/O, counting/timing and direct memory access—are handled by the established Z-80 Family peripherals Z-80 PIO, Z-80 SIO, Z-80 DART, Z-80 CTC and Z-80 DMA.

Easily programmed, the dual-channel **Z-80 PIO Parallel Input/Output Controller** offers two 8-bit I/O ports with individual handshake and pattern-recognition logic. Both ports operate in either byte or bit modes. In addition, interrupts can be programmed for peripheral status conditions and eight outputs are capable of driving Darlington transistors.

All common data communications protocols, asynchronous as well as synchronous, are remarkably well handled by the **Z-80 SIO Serial Input/Output Controller**. This dual-channel receiver/transmitter also offers on-chip parity and CRC generation/check-

ing, FIFO buffering, as well as flag and frame detection/generation logic.

Asynchronous-only applications are equally well handled by the **Z-80 DART Dual Asynchronous Receiver/Transmitter**. This cost-effective device offers all Z-80 SIO asynchronous-mode features in two channels.

Timing and event-counting functions are the forte of the **Z-80 CTC Counter/Timer Controller**. Its four counters all have individually programmable prescalers. Three counter outputs can drive Darlington transistors. The CTC is a convenient source of programmable clock rates for the SIO.

Data can be transferred directly between any two ports (I/O and memory, typically) with the

The DMA transfers, searches or search/transfers data in byte-at-a-time, burst or continuous modes. This device can achieve an impressive 2M bytes per second data rate in the search mode.

Three New Members. The popular, well established Z-80 Family of peripherals is complemented by three new controllers*. The Z-CIO/U, Z-UPC/U and Z-FIO feature sophisticated counting/timing and parallel I/O on one chip, off-line CPU processing and asynchronous I/O interfacing.

Although they function smoothly in most Z-80 systems, these controllers are not dedicated to the Z-80 bus. Instead, they fit a general-purpose bus that allows them to be incorporated into most 8-bit microprocessor systems.

In addition to its three independent 16-bit counters, the **Z-CIO/U Counter and Input/Output Controller** offers two 8-bit bidirectional I/O ports plus a special-purpose 4-bit port. Its four handshake modes include IEEE-488. Furthermore, the CIO pattern recognition logic doubles as a 16-input interrupt controller.

Z-UPC/U Universal Peripheral Controller solves general-purpose control and data-manipulation problems. This complete microcomputer-on-a-chip executes the same friendly, capable instruction set as the Z8 MCU Microcomputer. The UPC also offers 2K of on-chip ROM, three I/O ports and two programmable 8-bit counter/timers with 6-bit prescalers.

Bits and pieces of asynchronous parallel-processing systems are interconnected with the **Z-FIO FIFO Input/Output Interface Unit**. This surprisingly flexible device can interface any major microprocessor and most peripherals to a system bus.

The Zilog 8-bit Microprocessor Family is the most popular microcomputer-component family in the world. Now you know why.

Z-80® CPU Central Processing Unit



**NEW
1980**

Product Brief

1

February 1980

Features

- 158 instructions, software compatible with all 78 of the 8080A instructions.
- High speed—up to 4 MHz rate with a minimum 1.0 μ s instruction execution time for the Z-80A and 2.5 MHz and 1.5 μ s for the Z-80.
- Duplicate set of general-purpose and flag registers (16 total) eases design and operation of such system software programming, single context switching, background-foreground programming, single-level interrupt processing. Two 16-bit index registers allow efficient processing of tables and arrays.
- Extensive instruction set includes string, bit, byte, word operations and block transfers along with efficient addressing modes such as indexed and relative.
- All Z-80 Family components offer bus-request and prioritized interrupt request daisy chains that can be implemented without external logic.
- Three modes of high-speed interrupt processing: 8080 compatible, non-Z-80 peripheral and Z-80 Family peripheral daisy chain.

General Description

The Zilog Z-80 microprocessor is the central element of the Z-80 Microprocessor Family, which includes a variety of compatible peripheral circuits. The Z-80 Family handles most microprocessor applications with little additional logic. Z-80 designs are efficient and cost effective microcomputer systems.

The Z-80 CPU is a third generation 8-bit microprocessor with exceptional capabilities and computational power. The powerful

instruction set, dual register sets and other features of the Z-80 CPU provide higher throughput and greater efficiency in memory utilization. In addition, the Z-80 CPU is easy to use in a system because of its single +5 V supply requirement and fully compatible family of peripheral circuits. All Z-80 Family members are fabricated with scaled n-channel silicon-gate depletion-load technology.

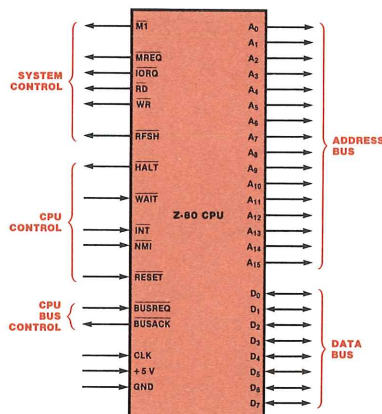


Figure 1. Pin Functions

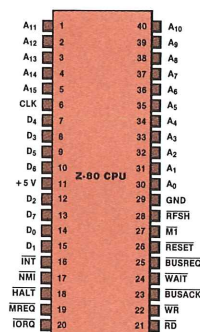


Figure 2. Pin Assignments

**General
Description**
(Continued)

The internal register configuration contains 208 bits of Read/Write memory accessible to the programmer. The registers include two sets of six general-purpose registers that may be used individually as 8-bit registers or as 16-bit register pairs. There are also two sets of accumulator and flag registers.

The programmer has access to either set of main or alternate registers through a group of exchange instructions. This alternate set allows foreground/background mode of operation or may be reserved for very fast interrupt response. Each CPU also contains a 16-bit stack pointer which permits simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data handling.

The two 16-bit index registers allow tabular data manipulation and easy implementation of relocatable code. The refresh register provides for automatic, totally transparent refresh of external dynamic memories. The I register is used in a powerful interrupt response mode to form the upper eight bits of a pointer to an interrupt service address table, while the interrupting device supplies the lower eight bits of the pointer. An indirect call is then made to this service address.

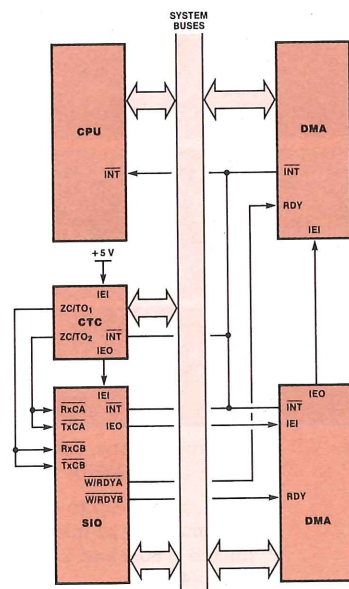


Figure 3. Typical Z-80 Family Environment

**Instruction
Set**

The Z-80 CPU has one of the most powerful and rich instruction sets available in any 8-bit microprocessor. It includes such unique operations as block move for fast and efficient data

transfers between memory and I/O and bit operations on any location in memory. The Z-80 CPU offers instructions in the following categories:

8-bit loads
16-bit loads
Register exchanges
Memory block moves
Memory block searches
8-bit arithmetic and logic operations
16-bit arithmetic operations
Rotates and shifts

General-purpose Accumulator and Flag operations
Bit set, reset and test operations
Input and output operations
Jumps
Calls
Restarts
Returns

Z-80® PIO Parallel Input/Output Controller



**NEW
1980**

Product Brief

1

February 1980

Features

- Provides a direct interface between Z-80 microcomputer systems and peripheral devices.
- Both ports have interrupt-driven handshake for fast response.
- Four programmable operating modes: byte input, byte output, byte input/output (Port A only), and bit input/output.
- Programmable interrupts on peripheral status conditions.
- Standard Z-80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic.
- The eight Port B outputs can drive Darlington transistors (1.5 mA at 1.5 V).

General Description

The Z-80 PIO Parallel I/O Circuit is a programmable, dual-port device that provides a TTL-compatible interface between peripheral devices and the Z-80 CPU. The CPU configures the Z-80 PIO to interface with a wide range of peripheral devices with no other external logic. Typical peripheral devices that are compatible with the Z-80 PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc.

One characteristic of the Z-80 PIO that separates it from other interface controllers is that all data transfer between the peripheral device and the CPU is accomplished under

interrupt control. The interrupt logic of the PIO permits full use of the efficient interrupt capabilities of the Z-80 CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO.

Another feature is that the PIO can be programmed to interrupt the CPU on the occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the time the processor must spend in polling peripheral status.

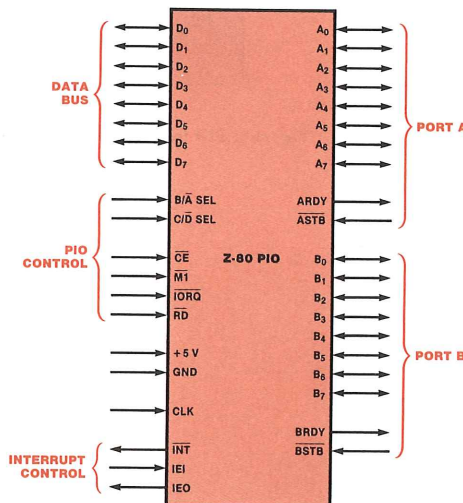


Figure 1. Pin Functions

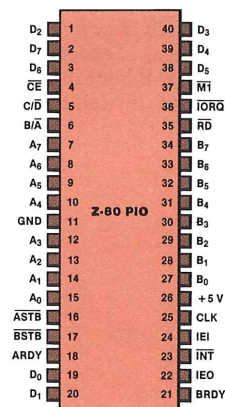


Figure 2. Pin Assignments

General Description (Continued)

The Z-80 PIO interfaces to peripherals via two independent general-purpose I/O ports, designated Port A and Port B. Each port has eight data bits and two handshake signals, Ready and Strobe, which control data transfer. The Ready output indicates to the peripheral that the port is ready for a data transfer. Strobe is an input from the peripheral that indicates a data transfer has occurred.

Operating Modes. The Z-80 ports can be programmed to operate in four modes: byte output (Mode 0), byte input (Mode 1), byte input/output (Mode 2) and bit input/output (Mode 3).

In Mode 0, either Port A or Port B can be programmed to output data. Both ports have output registers that are individually addressed by the CPU; data can be written to either port at any time. When data is written to a port, an active Ready output indicates to the external device that data is available at the associated port, and is ready for transfer to the external device. After the data transfer, the external device responds with an active Strobe input, which generates an interrupt, if enabled.

In Mode 1, either Port A or Port B can be configured in the input mode. Each port has an input register addressed by the CPU. When the CPU reads data from a port, the PIO sets the Ready signal, which is sensed by the external device. The external device then places data on the I/O lines and strobos the I/O port, which latches the data into the Port Input Register, resets Ready, and triggers the Interrupt Request, if enabled. The CPU can read the input data at any time, which again sets Ready.

Mode 2 is bidirectional and uses Port A, plus the interrupts and handshake signals from both ports. Port B must be set to Mode 3, and must be masked off. In operation, Port A is used for both data input and output. Output operation is similar to Mode 0 except data is allowed out onto the Port A bus only when $\overline{\text{ASTB}}$ is Low. For input, operation is similar to Mode 1, except that the data input uses the Port B handshake signals and the Port B interrupt (if enabled).

Both ports can be used in Mode 3. In this mode, the individual bits are defined as either input or output bits. This provides up to eight separate, individually defined bits for each port. During operation, Ready and Strobe are not used. Instead, an interrupt is generated if the condition of one input changes, or if all inputs change. The requirements for generating an interrupt are defined during the programming operation; the active level is specified as either High or Low, and the logic condition is specified as either one input active (OR), or all inputs active (AND). For example, if the port is programmed for active Low inputs, and the logic function is AND, then all inputs at the specified port must go Low to generate an interrupt.

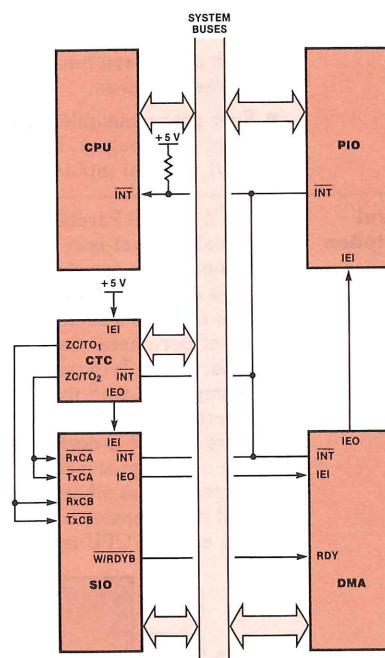


Figure 3. PIO in a Typical Z80 Family Environment

Z-80[®] CTC Counter/Timer Circuit



NEW
1980

Product Specification

1

February 1980

Features

- Four independently programmable counter/timer channels, each with a readable downcounter and a selectable 16 or 256 prescaler. Downcounters are reloaded automatically at zero count.
- Standard Z-80 Family daisy-chain interrupt structure provides fully vectored, prioritized interrupts without external logic. The CTC may also be used as an interrupt controller.
- Selectable positive or negative trigger initiates timer operation.
- Three channels have Zero Count/Timeout outputs capable of driving Darlington transistors.
- Interfaces directly to the Z-80 CPU or—for baud rate generation—to the Z-80 SIO.

Description

The Z-80 CTC four-channel counter/timer can be programmed by system software for a broad range of counting and timing applications. The four independently programmable channels of the Z-80 CTC satisfy common microcomputer system requirements for event counting, interrupt and interval timing, and general clock rate generation.

System design is simplified because the CTC connects directly to both the Z-80 CPU and the Z-80 SIO with no additional logic. In larger systems, address decoders and buffers may be required.

Programming the CTC is straightforward:

each channel is programmed with two bytes; a third is necessary when interrupts are enabled. Once started, the CTC counts down, reloads its time constant automatically, and resumes counting. Software timing loops are completely eliminated. Interrupt processing is simplified because only one vector need be specified; the CTC internally generates a unique vector for each channel.

The Z-80 CTC requires a single +5 V power supply and the standard Z-80 single-phase system clock. It is fabricated with n-channel silicon-gate depletion-load technology, and packaged in a 28-pin plastic or ceramic DIP.

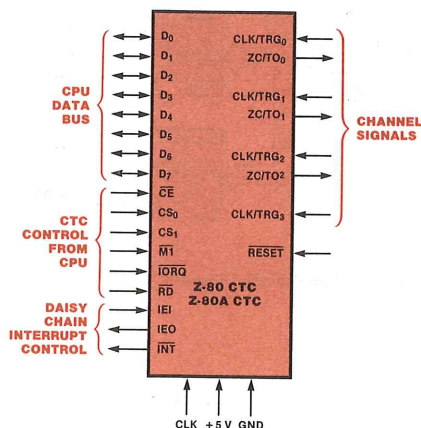


Figure 1. Pin Functions

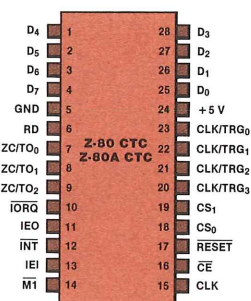


Figure 2. Pin Assignments

Functional Description

The Z-80 CTC has four independent counter/timer channels. Each channel is individually programmed with two words: a control word and a time-constant word. The control word selects the operating mode (counter or timer), enables or disables the channel interrupt, and selects certain other operating parameters. If the timing mode is selected, the control word also sets a prescaler, which divides the system clock by either 16 or 256. The time-constant word is a value from 1 to 256.

During operation, the individual counter channel counts down from the preset time constant value. In counter mode operation the counter decrements on each of the CLK/TRG input pulses until zero count is reached. Each decrement is synchronized by the system clock. For counts greater than 256, more than one counter can be cascaded. At zero count, the down-counter is automatically reset with the time constant value.

The timer mode determines time intervals as small as $4\ \mu\text{s}$ (Z-80A) or $6.4\ \mu\text{s}$ (Z-80) without additional logic or software timing loops. Time intervals are generated by dividing the system clock with a prescaler that decrements

a preset down-counter.

Thus, the time interval is an integral multiple of the clock period, the prescaler value (16 or 256) and the time constant that is preset in the down-counter. A timer is triggered automatically when its time constant value is programmed, or by an external CLK/TRG input.

Three channels have two outputs that occur at zero count. The first output is a zero-count/timeout pulse at the ZC/TO output. The fourth channel (Channel 3) does not have a ZC/TO output; interrupt request is the only output available from Channel 3.

The second output is Interrupt Request (INT), which occurs if the channel has its interrupt enabled during programming. When the Z-80 CPU acknowledges Interrupt Request, the Z-80 CTC places an interrupt vector on the data bus.

The four channels of the Z-80 CTC are fully prioritized and fit into four contiguous slots in a standard Z-80 daisy-chain interrupt structure. Channel 0 is the highest priority and Channel 3 the lowest. Interrupts can be individually enabled (or disabled) for each of the four channels.

Architecture

The CTC has four major elements, as shown in Figure 3.

- CPU bus I/O
- Channel control logic
- Interrupt logic
- Counter/timer circuits

CPU Bus I/O. The CPU bus I/O circuit decodes the address inputs, and interfaces the CPU data and control signals to the CTC for distribution on the internal bus.

Internal Control Logic. The CTC internal control logic controls overall chip operating functions such as the chip enable, reset, and read/write logic.

Interrupt Logic. The interrupt control logic ensures that the CTC interrupts interface properly with the Z-80 CPU interrupt system. The logic controls the interrupt priority of the CTC as a function of the IEI signal. If IEI is High, the CTC has priority. During interrupt

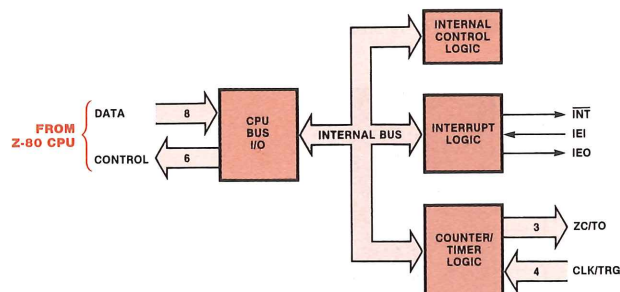


Figure 3. Functional Block Diagram

processing, the interrupt logic holds IEO Low, which inhibits the interrupt operation on lower priority devices. If the IEI input goes Low, priority is relinquished and the interrupt logic drives IEO Low.

If a channel is programmed to request an interrupt, the interrupt logic drives IEO Low at the zero count, and generates an INT signal to the Z-80 CPU. When the Z-80 CPU responds with interrupt acknowledge (MI and IORQ), then the interrupt logic arbitrates the CTC internal priorities, and the interrupt control logic places a unique interrupt vector on the data bus.

If an interrupt is pending, the interrupt logic holds IEO Low. When the Z-80 CPU issues a Return From Interrupt (RETI) instruction, each peripheral device decodes the first byte (ED₁₆). If the device has a pending interrupt, it raises IEO (High) for one M1 cycle. This ensures that all lower priority devices can decode the entire RETI instruction and reset properly.

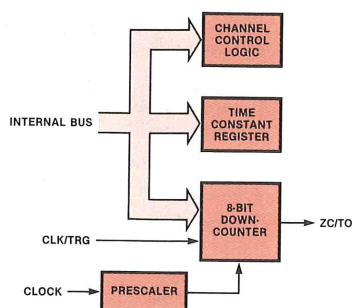


Figure 4. Counter/Timer Block Diagram

Counter/Timer Circuits. The CTC has four independent counter/timer circuits, each containing the logic shown in Figure 4.

Channel Control Logic. The channel control logic receives the 8-bit channel control word when the counter/timer channel is programmed. The channel control logic decodes

the control word and sets the following operating conditions:

- Interrupt enable (or disable)
- Operating mode (timer or counter)
- Timer mode prescaler factor (16 or 256)
- Active slope for CLK/TRG input
- Timer mode trigger (automatic or CLK/TRG input)
- Time constant data word to follow
- Software reset

Time Constant Register. When the counter/timer channel is programmed, the time constant register receives and stores an 8-bit time constant value, which can be anywhere from 1 to 256 (0 = 256). This constant is automatically loaded into the down-counter when the counter/timer channel is initialized, and subsequently after each zero count.

Prescaler. The prescaler, which is used only in timer mode, divides the system clock frequency by a factor of either 16 or 256. The prescaler output clocks the down-counter during timer operation. The effect of the prescaler on the down-counter is a multiplication of the system clock period by 16 or 256. The prescaler factor is programmed by bit 5 of the channel control word.

Down-Counter. Prior to each count cycle, the down-counter is loaded with the time constant register contents. The counter is then decremented one of two ways, depending on operating mode:

- By the prescaler output (timer mode)
- By the trigger pulses into the CLK/TRG input (counter mode)

Without disturbing the down-count, the Z-80 CPU can read the count remaining at any time by performing an I/O read operation at the port address assigned to the CTC channel. When the down-counter reaches the zero count, the ZC/TO output generates a positive-going pulse. When the interrupt is enabled, zero count also triggers an interrupt request signal (INT) from the interrupt logic.

Programming

Each Z-80 CTC channel must be programmed prior to operation. Programming consists of writing two words to the I/O port that corresponds to the desired channel. The first word is a control word that selects the operating mode and other parameters; the second word is a time constant, which is a binary data word with a value from 1 to 256. A time constant word must be preceded by a channel control word.

After initialization, channels may be reprogrammed at any time. If updated control and time constant words are written to a channel during the count operation, the count continues to zero before the new time constant is loaded into the counter.

If the interrupt on any Z-80 CTC channel is enabled, the programming procedure should also include an interrupt vector. Only one vector is required for all four channels, because the interrupt logic automatically modifies the vector for the channel requesting service.

A control word is identified by a 1 in bit 0. A 0 in bit 2 indicates a time constant word is to follow. Interrupt vectors are always addressed to Channel 0, and identified by a 0 in bit 0.

Addressing. During programming, channels are addressed with the channel select pins CS₁ and CS₂. A 2-bit binary code selects the appropriate channel as shown in the following table.

Channel	CS ₁	CS ₀
0	0	0
1	0	1
2	1	0
3	1	1

Reset. The CTC has both hardware and software resets. The hardware reset terminates all down-counts and disables all CTC interrupts by resetting the interrupt bits in the control registers. In addition, the ZC/TO and Interrupt outputs go inactive, IEO reflects IEL, and

D₀-D₇ go to the high-impedance state. All channels must be completely reprogrammed after a hardware reset.

The software reset is controlled by bit 1 in the channel control word. When a channel receives a software reset, it stops counting. When a software reset is used, the other bits in the control word also change the contents of the channel control register. After a software reset a new time constant word must be written to the same channel.

If the channel control word has both bits D₁ and D₂ set to 1, the addressed channel stops operating, pending a new time constant word. The channel is ready to resume after the new constant is programmed. In timer mode, if D₃ = 0, operation is triggered automatically when the time constant word is loaded.

Channel Control Word Programming. The channel control word is shown in Figure 5. It sets the modes and parameters described below.

Interrupt Enable. D₇ enables the interrupt, so that an interrupt output (INT) is generated at zero count. Interrupts may be programmed in either mode and may be enabled or disabled at any time.

Operating Mode. D₆ selects either timer or counter mode.

Prescaler Factor. (Timer Mode Only). D₅ selects factor—either 16 or 256.

Trigger Slope. D₄ selects the active edge or slope of the CLK/TRG input pulses. Note that reprogramming the CLK/TRG slope during operation is equivalent to issuing an active edge. If the trigger slope is changed by a control word update while a channel is pending operation in timer mode, the result is the same as a CLK/TRG pulse and the timer starts. Similarly, if the channel is in counter mode, the counter decrements.

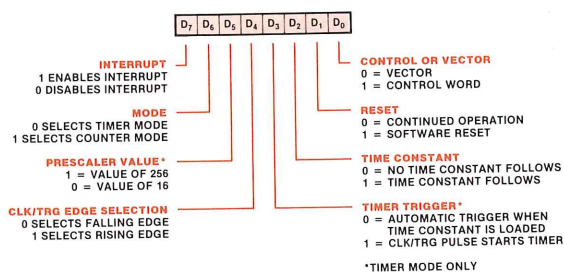


Figure 5. Channel Control Word

Programming (Continued)

Trigger Mode (Timer Mode Only). D_3 selects the trigger mode for timer operation. When D_3 is reset to 0, the timer is triggered automatically. The time constant word is programmed during an I/O write operation, which takes one machine cycle. At the end of the write operation there is a setup delay of one clock period. The timer starts automatically (decrements) on the rising edge of the second clock pulse (T_2) of the machine cycle following the write operation. Once started, the timer runs continuously. At zero count the timer reloads automatically and continues counting without interruption or delay, until stopped by a reset.

When D_3 is set to 1, the timer is triggered externally through the CLK/TRG input. The time constant word is programmed during an I/O write operation, which takes one machine cycle. The timer is ready for operation on the rising edge of the second clock pulse (T_2) of the following machine cycle. Note that the first timer decrement follows the active edge of the CLK/TRG pulse by a delay time of one clock cycle if a minimum setup time to the rising edge of clock is met. If this minimum is not met, the delay is extended by another clock period. Consequently, for immediate triggering, the CLK/TRG input must precede T_2 by one clock cycle plus its minimum setup time. If the minimum time is not met, the timer will start on the third clock cycle (T_3).

Once started the timer operates continuously, without interruption or delay, until stopped by a reset.

Time Constant to Follow. A 1 in D_2 indicates that the next word addressed to the selected channel is a time constant data word for the time constant register. The time constant word may be written at any time.

A 0 in D_2 indicates no time constant word is to follow. This is ordinarily used when the channel is already in operation and the new channel control word is an update. A channel will not operate without a time constant value. The only way to write a time constant value is to write a control word with D_2 set.

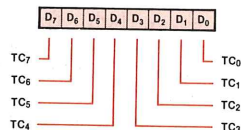


Figure 6. Time Constant Word

Software Reset. Setting D_1 to 1 causes a software reset, which is described in the Reset section.

Control Word. Setting D_0 to 0 identifies the word as a control word.

Time Constant Programming. Before a channel can start counting it must receive a time constant word from the CPU. During programming or reprogramming, a channel control word in which bit 2 is set must precede the time constant word to indicate that the next word is a time constant. The time constant word can be any value from 1 to 256 (Figure 6). Note that 00_{16} is interpreted as 256.

In timer mode, the time interval is controlled by three factors:

- The system clock period (ϕ)
- The prescaler factor (P), which multiplies the interval by either 16 or 256
- The time constant (T), which is programmed into the time constant register

Consequently, the time interval is the product of $\phi \times P \times T$. The minimum timer resolution is $16 \times \phi$ ($4 \mu s$ with a 4 MHz clock). The maximum timer interval is $256 \times \phi \times 256$ (16.4 ms with a 4 MHz clock). For longer intervals timers may be cascaded.

Interrupt Vector Programming. If the Z-80 CTC has one or more interrupts enabled, it can supply interrupt vectors to the Z-80 CPU. To do so, the Z-80 CTC must be pre-programmed with the most-significant five bits of the interrupt vector. Programming consists of writing a vector word to the I/O port corresponding to the Z-80 CTC Channel 0. Note that D_0 of the vector word is always zero, to distinguish the vector from a channel control word. D_1 and D_2 are not used in programming the vector word. These bits are supplied by the interrupt logic to identify the channel requesting interrupt service with a unique interrupt vector (Figure 7). Channel 0 has the highest priority.

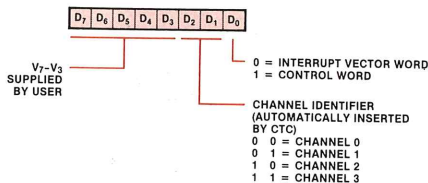


Figure 7. Interrupt Vector Word

**Pin
Description**

$\overline{CE}$. *Chip Enable* (input, active Low). When enabled the CTC accepts control words, interrupt vectors, or time constant data words from the data bus during an I/O write cycle; or transmits the contents of the down-counter to the CPU during an I/O read cycle. In most applications this signal is decoded from the eight least significant bits of the address bus for any of the four I/O port addresses that are mapped to the four counter-timer channels.

CLK. *System Clock* (input). Standard single-phase Z-80 system clock.

CLK/TRG₀-CLK/TRG₃. *External Clock/Timer Trigger* (input, user-selectable active High or Low). Four pins corresponding to the four Z-80 CTC channels. In counter mode, every active edge on this pin decrements the down-counter. In timer mode, an active edge starts the timer.

CS₀-CS₁. *Channel Select* (inputs active High). Two-bit binary address code selects one of the four CTC channels for an I/O write or read (usually connected to A₀ and A₁).

D₀-D₇. *System Data Bus* (bidirectional, 3-state). Transfers all data and commands between the Z-80 CPU and the Z-80 CTC.

IEI. *Interrupt Enable In* (input, active High). A High indicates that no other interrupting devices of higher priority in the daisy chain are being serviced by the Z-80 CPU.

IEO. *Interrupt Enable Out* (output, active High). High only if IEI is High and the Z-80 CPU is not servicing an interrupt from any Z-80 CTC channel. IEO blocks lower priority devices from interrupting while a higher priority interrupting device is being serviced.

INT. *Interrupt Request* (output, open drain, active Low). Low when any Z-80 CTC channel that has been programmed to enable interrupts has a zero-count condition in its down-counter.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). Used with $\overline{CE}$ and RD to transfer data and channel control words between the Z-80 CPU and the Z-80 CTC. During a write cycle, $\overline{IORQ}$ and $\overline{CE}$ are active and RD inactive. The Z-80 CTC does not receive a specific write signal; rather, it internally generates its own from the inverse of an active RD signal. In a read cycle, $\overline{IORQ}$, $\overline{CE}$ and RD are active; the contents of the down-counter are read by the Z-80 CPU. If $\overline{IORQ}$ and M1 are both true, the CPU is acknowledging an interrupt request, and the highest priority interrupting channel places its interrupt vector on the Z-80 data bus.

M1. *Machine Cycle One* (input from CPU, active Low). When M1 and $\overline{IORQ}$ are active, the Z-80 CPU is acknowledging an interrupt. The Z-80 CTC then places an interrupt vector on the data bus if it has highest priority, and if a channel has requested an interrupt (INT).

RD. *Read Cycle Status* (input, active Low). Used in conjunction with $\overline{IORQ}$ and $\overline{CE}$ to transfer data and channel control words between the Z-80 CPU and the Z-80 CTC.

RESET. *Reset* (input active Low). Terminates all down-counts and disables all interrupts by resetting the interrupt bits in all control registers; the ZC/TO and the Interrupt outputs go inactive; IEO reflects IEI; D₀-D₇ go to the high-impedance state.

ZC/TO₀-ZC/TO₂. *Zero Count/Timeout* (output, active High). Three ZC/TO pins corresponding to Z-80 CTC channels 2 through 0 (Channel 3 has no ZC/TO pin). In both counter and timer modes the output is an active High pulse when the down-counter decrements to zero.

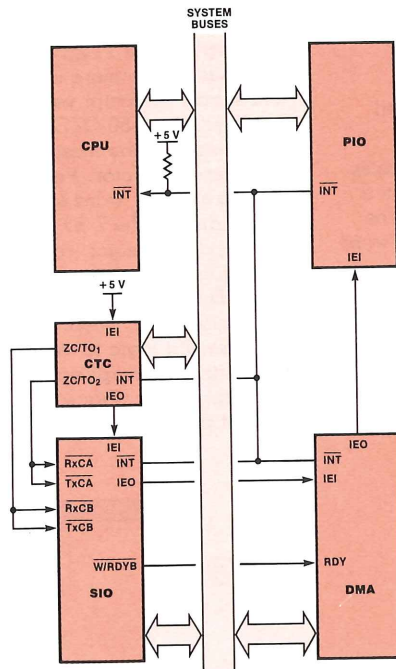


Figure 8. A Typical Z-80 Environment

Timing

Read Cycle Timing. Figure 9 shows read cycle timing. This cycle reads the contents of a down-counter without disturbing the count. During clock cycle T_2 , the Z-80 CPU initiates a read cycle by driving the following inputs Low: $\overline{RD}$, $\overline{IORQ}$, and $\overline{CE}$. A 2-bit binary code at inputs CS_1 and CS_0 selects the channel to be read. $\overline{M1}$ must be High to distinguish this cycle from an interrupt acknowledge. No additional wait states are allowed.

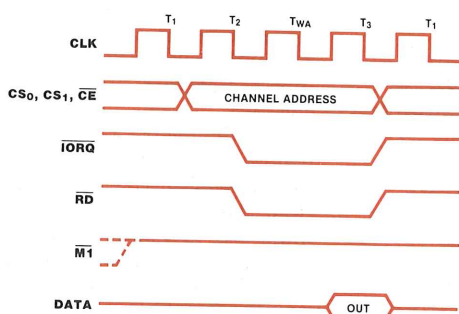


Figure 9. Read Cycle Timing

Write Cycle Timing. Figure 10 shows write cycle timing for loading control, time constant or vector words.

The CTC does not have a write signal input, so it generates one internally when the read ($\overline{RD}$) input is High during T_1 . During T_2 $\overline{IORQ}$ and $\overline{CE}$ inputs are Low. $\overline{M1}$ must be High to distinguish a write cycle from an interrupt acknowledge. A 2-bit binary code at inputs CS_1 and CS_0 selects the channel to be addressed, and the word being written is placed on the Z-80 data bus. The data word is

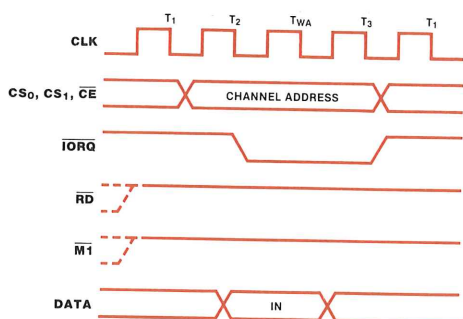


Figure 10. Write Cycle Timing

latched into the appropriate register with the rising edge of clock cycle T_{WA} . No additional wait states are allowed.

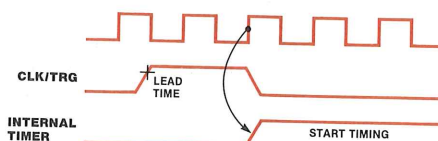


Figure 11. Timer Mode Timing

Timer Operation. In the timer mode, a CLK/TRG pulse input starts the timer (Figure 11) on the second succeeding rising edge of CLK. The trigger pulse is asynchronous, and it must have a minimum width. A minimum lead time (210 ns) is required between the active edge of the CLK/TRG and the next rising edge of CLK to enable the prescaler on the following clock edge. If the CLK/TRG edge occurs closer than this, the initiation of the timer function is delayed one clock cycle. This corresponds to the startup timing discussed in the programming section. The timer can also be started automatically if so programmed by the channel control word.

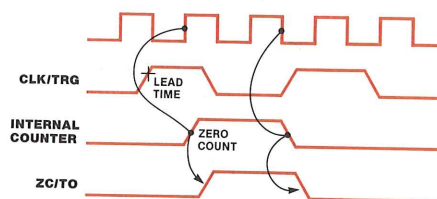


Figure 12. Counter Mode Timing

Counter Operation. In the counter mode, the CLK/TRG pulse input decrements the down-counter. The trigger is asynchronous, but the count is synchronized with CLK. For the decrement to occur on the next rising edge of CLK, the trigger edge must precede CLK by a minimum lead time as shown in Figure 12. If the lead time is less than specified, the count is delayed by one clock cycle. The trigger pulse must have a minimum width, and the trigger period must be at least twice the clock period.

The ZC/TO output occurs immediately after zero count, and follows the rising CLK edge.

Interrupt Operation

The Z-80 CTC follows the Z-80 system interrupt protocol for nested priority interrupts and return from interrupt, wherein the interrupt priority of a peripheral is determined by its location in a daisy chain. Two lines—IEI and IEO—in the CTC connect it to the system daisy chain. The device closest to the +5 V supply has the highest priority (Figure 13). For additional information on the Z-80 interrupt structure, refer to the *Z-80 CPU Product Specification* and the *Z-80 CPU Technical Manual*.

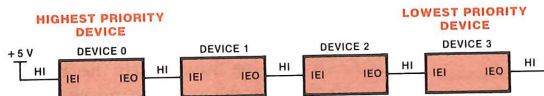


Figure 13. Daisy-Chain Interrupt Priorities

Within the Z-80 CTC, interrupt priority is predetermined by channel number: Channel 0 has the highest priority, and Channel 3 the lowest. If a device or channel is being serviced with an interrupt routine, it cannot be interrupted by a device or channel with lower priority until service is complete. Higher priority devices or channels may interrupt the servicing of lower priority devices or channels.

A Z-80 CTC channel may be programmed to request an interrupt every time its down-counter reaches zero. Note that the CPU must be programmed for interrupt mode 2. Some time after the interrupt request, the CPU sends an interrupt acknowledge. The CTC interrupt control logic determines the highest priority channel that is requesting an interrupt. Then, if the CTC IEI input is High (indicating that it has priority within the system daisy chain) it places an 8-bit interrupt vector on the system data bus. The high-order five bits of this vector

were written to the CTC during the programming process; the next two bits are provided by the CTC interrupt control logic as a binary code that identifies the highest priority channel requesting an interrupt; the low-order bit is always zero.

Interrupt Acknowledge Timing. Figure 14 shows interrupt acknowledge timing. After an interrupt request, the Z-80 CPU sends an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). All channels are inhibited from changing their interrupt request status when $\overline{M1}$ is active—about two clock cycles earlier than $\overline{IORQ}$. $\overline{RD}$ is High to distinguish this cycle from an instruction fetch.

The CTC interrupt logic determines the highest priority channel requesting an interrupt. If the CTC interrupt enable input (IEI) is High, the highest priority interrupting channel within the CTC places its interrupt vector on the data bus when $\overline{IORQ}$ goes Low. Two wait states (T_{WA}) are automatically inserted at this time to allow the daisy chain to stabilize. Additional wait states may be added.

Return from Interrupt Timing. At the end of an interrupt service routine the RETI (Return From Interrupt) instruction initializes the daisy chain enable lines for proper control of nested priority interrupt handling. The CTC decodes the 2-byte RETI code internally and determines whether it is intended for a channel being serviced. Figure 15 shows RETI timing.

If several Z-80 peripherals are in the daisy chain, IEI settles active (High) on the chip currently being serviced when the opcode ED_{16} is decoded. If the following opcode is $4D_{16}$, the peripheral being serviced is released and its IEO becomes active. Additional wait states are allowed.

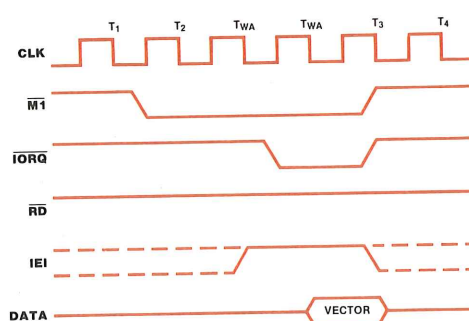


Figure 14. Interrupt Acknowledge Timing

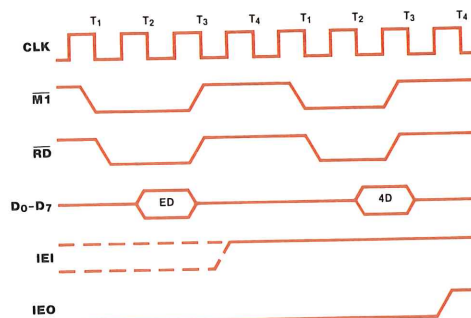


Figure 15. Return From Interrupt Timing

Absolute Maximum Ratings

Voltages on any pin with respect to GND -0.3 V to +7.0 V
Operating Ambient Temperature . . 0°C to 70°C
Storage Temperature -65°C to +150°C

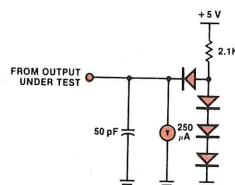
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75 \text{ V} \leq V_{CC} \leq +5.25 \text{ V}$
- $\text{GND} = 0 \text{ V}$
- $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$
- Output Test Circuit

All ac parameters assume a load capacitance of 50 pF max.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	$V_{CC} - .6$	$V_{CC} + .3$	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	V_{CC}	V	$I_{OL} = 2 \text{ mA}$
V_{OL}	Output Low Voltage		+0.4	V	$I_{OH} = 250 \mu\text{A}$
V_{OH}	Output High Voltage	+2.4		V	$V_{IN} = 0 \text{ to } V_{CC}$
I_{CC}	Power Supply Current		+120	mA	$V_{OUT} = 2.4 \text{ to } V_C$
I_{LI}	Input Leakage Current		+10	μA	$V_{OUT} = 0.4 \text{ V}$
I_{LOH}	3-State Output Leakage Current in Float		+10	μA	
I_{LOL}	3-State Output Leakage Current in Float		-10	μA	
I_{OHD}	Darlington Drive Current	-1.5		mA	$V_{OH} = 1.5 \text{ V}$ $R_{EXT} = 390\Omega$

Capacitance

Symbol	Parameter	Max	Unit	Condition
CLK	Clock Capacitance	20	pF	Unmeasured pins
C_{IN}	Input Capacitance	5	pF	returned to ground
C_{OUT}	Output Capacitance	10	pF	

$T_A = 25^\circ\text{C}$, $f = 1 \text{ MHz}$

AC Character- istics

Number	Symbol	Parameter	Z-80 CTC		Z-80A CTC		Comment
			Min ns	Max ns	Min ns	Max ns	
1	TcC	Clock Cycle Time	400	[1]	250	[1]	
2	TwCH	Clock Width (High)	170	2000	105	2000	
3	TwCl	Clock Width (Low)	170	2000	105	2000	
4	TfC	Clock Fall Time		30		30	
5	TrC	Clock Rise Time		30		30	
6	Th	All Hold Times	0		0		
7	TsCS(C)	CS to Clock ↑ Setup Time	250		160		
8	TsCE(C)	$\overline{CE}$ to Clock ↑ Setup Time	200		150		
9	TsIO(C)	$\overline{IORQ}$ ↓ to Clock ↑ Setup Time	250		115		
10	TsRD(C)	$\overline{RD}$ ↓ to Clock ↑ Setup Time	240		115		
11	TdC(DO)	Clock ↑ to Data Out Delay		240		200	[2]
12	TdC(DOz)	Clock ↑ to Data Out Float Delay		230		110	
13	TsDI(C)	Data In to Clock ↑ Setup Time	60		50		
14	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (Interrupt immediately preceding $\overline{M1}$)		300		190	[3, 5]
16	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		340		160	[2]
17	TdIEI(IEOf)	IEI ↑ to IEO ↓ Delay		190		130	[3]
18	TdIEI(IEOr)	IEI ↑ to IEO ↓ Delay (After ED Decode)		220		160	[3]
19	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay		(TcC + 200)		(TcC + 140)	Timer mode
20	TdCLK(INT)	CLK/TRG ↑ to $\overline{INT}$ ↓ tsCTR(C) satisfied tsCTR(C) not satisfied		(TcC + 230) (2TcC + 530)		(TcC + 160) (2TcC + 370)	Counter mode Counter mode Counter mode
21	TcCTR	CLK/TRG Cycle Time	(2TcC)		(2TcC)		
22	TrCTR	CLK/TRG Rise Time		50		50	
23	TfCTR	CLK/TRG Fall Time		50		50	
24	TwCTRl	CLK/TRG Width (Low)	200		200		
25	TwCTRh	CLK/TRG Width (High)	200		200		
26	TsCTR(Cs)	CLK/TRG ↑ to Clock ↑ Setup Time for Immediate Count	300		210		Counter mode
27	TsCTR(Ct)	CLK/TRG ↑ to Clock ↑ Setup Time for enabling of Prescaler on following clock↑	210		210		Timer mode
28	TdC(ZC/TOr)	Clock ↑ to ZC/TO ↓ Delay		260		190	
29	TdC(ZC/TOf)	Clock ↓ to ZC/TO ↓ Delay		190		190	

NOTES:

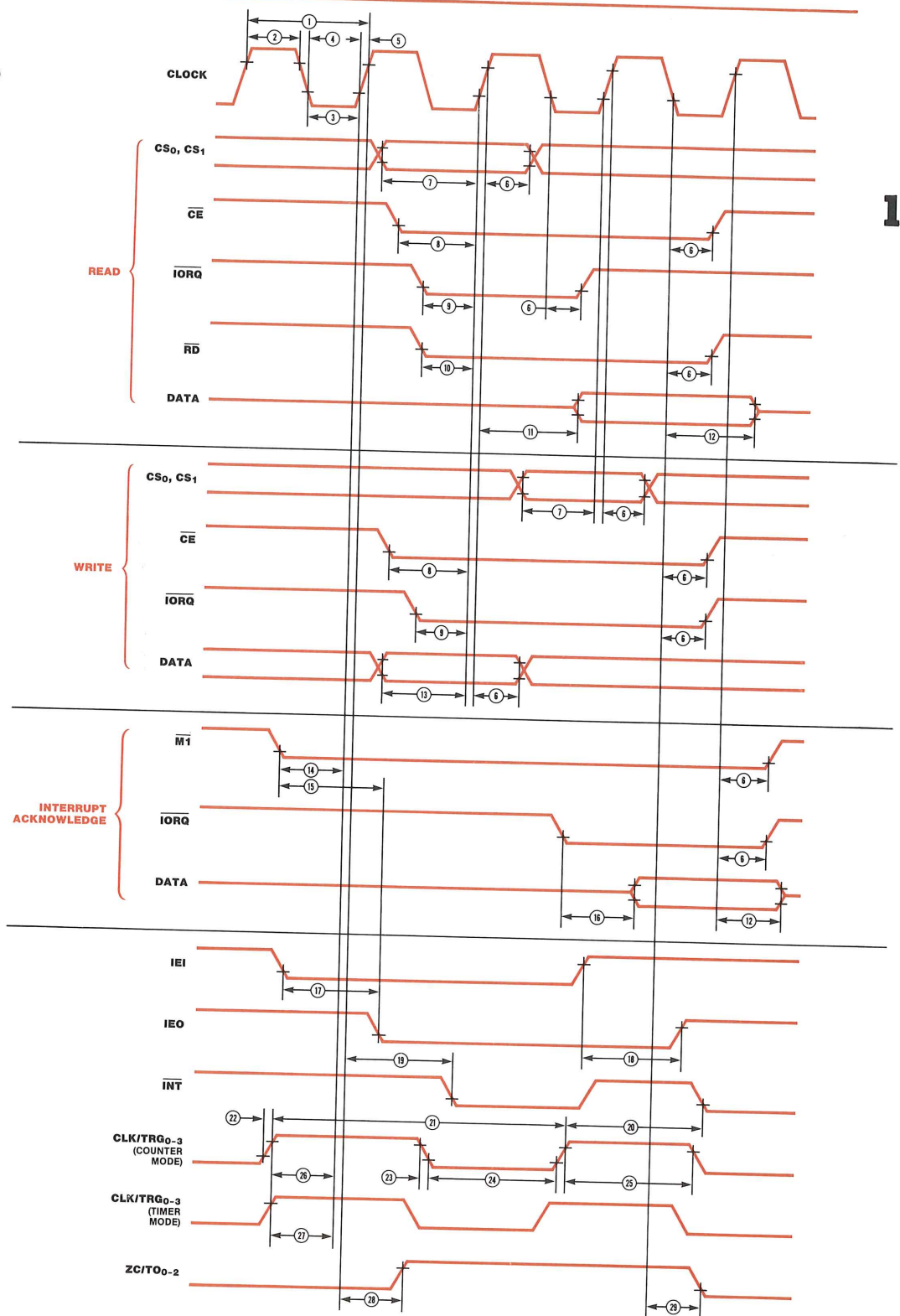
[1] $TcC = TwCh + TwCl + TrC + TfC$

[2] Increase delay by 10 ns for each 50 pF increase in loading, 200 pF maximum for data lines and 100 pF for control lines.

[3] Increase delay by 2 ns for each 10 pF increase in loading, 100 pF maximum.

[4] RESET must be active for a minimum of 3 clock cycles.

AC
Character-
istics
 (Continued)



Z-80[®] SIO Serial Input/Output Controller



NEW
1980

Product Specification

1

February 1980

Features

- Two independent full-duplex channels, with separate control and status lines for modems or other devices.
- Data rates of 0 to 500K bits/second in the x1 clock mode with a 2.5 MHz clock (Z-80 SIO), or 0 to 800K bits/second with a 4.0 MHz clock (Z-80A SIO).
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7 or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity; overrun and framing error detection.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7 or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection and flag insertion.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

General Description

The Z-80 SIO Serial Input/Output Controller is a dual-channel data communication interface with extraordinary versatility and capability. Its basic functions as a serial-to-parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or

bit-oriented, and performs all of the functions traditionally done by UARTs, USARTs and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this on two fully-independent channels, with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU or DMA

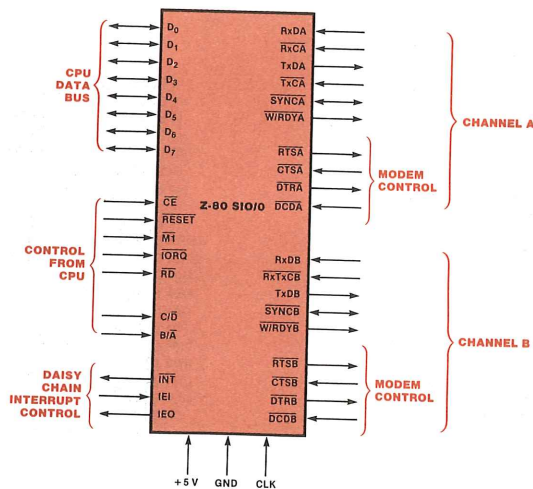


Figure 1. Z-80 SIO/0 Pin Functions

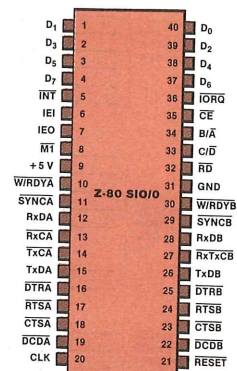


Figure 2. Z-80 SIO/0 Pin Assignments

General Description (Continued)

control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast (or slow) peripheral devices. While designed primarily as a member of the Z-80 family, its versatility makes it well suited to many other CPUs.

Pin Description

Figures 1 through 6 illustrate the three pin configurations (bonding options) available in the SIO. The constraints of a 40-pin package make it impossible to bring out the Receive Clock ($\overline{\text{RxC}}$), Transmit Clock ($\overline{\text{TxC}}$), Data Terminal Ready (DTR) and Sync (SYNC) signals for both channels. Therefore, either Channel B lacks a signal or two signals are bonded together in the three bonding options offered:

- Z-80 SIO/2 lacks $\overline{\text{SYNCB}}$
- Z-80 SIO/1 lacks $\overline{\text{DTRB}}$
- Z-80 SIO/0 has all four signals, but $\overline{\text{TxCB}}$ and $\overline{\text{RxCB}}$ are bonded together

The first bonding option above (SIO/2) is the preferred version for most applications. The pin descriptions are as follows:

B/ $\overline{\text{A}}$. Channel A Or B Select (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the SIO. Address bit A_0 from the CPU is often used for the selection function.

C/ $\overline{\text{D}}$. Control Or Data Select (input, High selects Control). This input defines the type of information transfer performed between the CPU and the SIO. A High at this input during a CPU write to the SIO causes the information on the data bus to be interpreted as a command for the channel selected by B/ $\overline{\text{A}}$. A Low at C/ $\overline{\text{D}}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

The Z-80 SIO is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V power supply and the standard Z-80 family single-phase clock.

$\overline{\text{CE}}$. Chip Enable (Input, active Low). A Low level at this input enables the SIO to accept command or data input from the CPU during a write cycle, or to transmit data to the CPU during a read cycle.

CLK. System Clock (input). The SIO uses the standard Z-80 System Clock to synchronize internal signals. This is a single-phase clock.

$\overline{\text{CTS}}$, $\overline{\text{CTS}}$. Clear To Send (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger buffering does not guarantee a specified noise-level margin.

D_0 - D_7 . System Data Bus (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z-80 SIO. D_0 is the least significant bit.

$\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$. Data Carrier Detect (inputs, active Low). These pins function as receiver enables if the SIO is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt-trigger buffer-

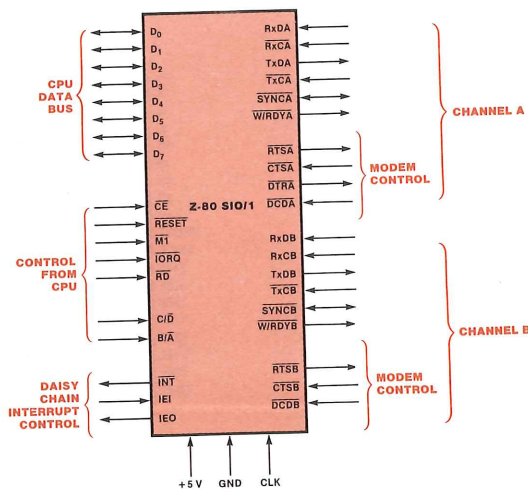


Figure 3. Z-80 SIO/1 Pin Functions

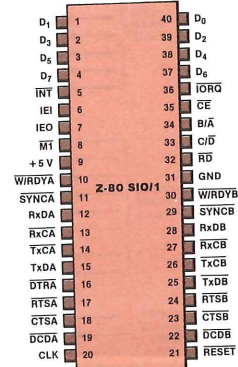


Figure 4. Z-80 SIO/1 Pin Assignments

Pin Description (Continued)

ing does not guarantee a specific noise-level margin.

DTRA, DTRB. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into Z-80 SIO. They can also be programmed as general-purpose outputs.

In the Z-80 SIO/1 bonding option, $\overline{\text{DTRB}}$ is omitted.

IEI. *Interrupt Enable In* (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

INT. *Interrupt Request* (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls $\overline{\text{INT}}$ Low.

IORQ. *Input/Output Request* (input from CPU, active Low). $\overline{\text{IORQ}}$ is used in conjunction with $\overline{\text{B/A}}$, $\overline{\text{C/D}}$, $\overline{\text{CE}}$ and $\overline{\text{RD}}$ to transfer commands and data between the CPU and the SIO. When $\overline{\text{CE}}$, $\overline{\text{RD}}$ and $\overline{\text{IORQ}}$ are all active, the channel selected by $\overline{\text{B/A}}$ transfers data to the CPU (a read operation). When $\overline{\text{CE}}$ and $\overline{\text{IORQ}}$ are active, but $\overline{\text{RD}}$ is inactive, the channel selected by $\overline{\text{B/A}}$ is written to by the CPU with either data or control information as specified by $\overline{\text{C/D}}$. As mentioned previously, if $\overline{\text{IORQ}}$ and

$\overline{\text{M1}}$ are active simultaneously, the CPU is acknowledging an interrupt and the SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

M1. *Machine Cycle* (input from Z-80 CPU, active Low). When $\overline{\text{M1}}$ is active and $\overline{\text{RD}}$ is also active, the Z-80 CPU is fetching an instruction from memory; when $\overline{\text{M1}}$ is active while $\overline{\text{IORQ}}$ is active, the SIO accepts $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ as an interrupt acknowledge if the SIO is the highest priority device that has interrupted the Z-80 CPU.

RxCA, RxCB. *Receiver Clocks* (inputs).

Receive data is sampled on the rising edge of $\overline{\text{RxC}}$. The Receive Clocks may be 1, 16, 32 or 64 times the data rate in asynchronous modes. These clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered (no noise level margin is specified).

In the Z-80 SIO/0 bonding option, $\overline{\text{RxCB}}$ is bonded together with $\overline{\text{TxCB}}$.

RD. *Read Cycle Status* (input from CPU, active Low). If $\overline{\text{RD}}$ is active, a memory or I/O read operation is in progress. $\overline{\text{RD}}$ is used with $\overline{\text{B/A}}$, $\overline{\text{CE}}$ and $\overline{\text{IORQ}}$ to transfer data from the SIO to the CPU.

RxDA, RxDB. *Receive Data* (inputs, active High). Serial data at TTL levels.

RESET. *Reset* (input, active Low). A Low RESET disables both receivers and transmitters, forces $\overline{\text{TxD}}$ and $\overline{\text{TxB}}$ marking, forces the modem controls High and disables all interrupts. The control registers must be

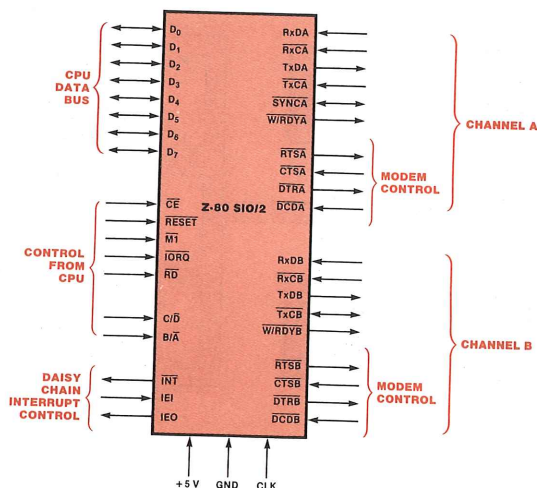


Figure 5. Z-80 SIO/2 Pin Functions

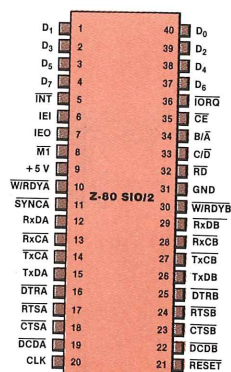


Figure 6. Z-80 SIO/2 Pin Assignments

**Pin
Description**
(Continued)

rewritten after the SIO is reset and before data is transmitted or received.

RTSA, RTSB. *Request To Send* (outputs, active Low). When the RTS bit in Write Register 5 (Figure 14) is set, the $\overline{\text{RTS}}$ output goes Low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the RTS pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

SYNCA, SYNCB. *Synchronization* (inputs/outputs, active Low). These pins can act either as inputs or outputs. In the asynchronous receive mode, they are inputs similar to CTS and DCD. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in Read Register 0 (Figure 13), but have no other function. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, SYNC must be driven Low on the second rising edge of $\overline{\text{RxC}}$ after that rising edge of $\overline{\text{RxC}}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the SYNC input. Once SYNC is forced Low, it should be kept Low until the CPU informs the external synchronization detect logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{\text{RxC}}$ that immediately precedes the falling edge of SYNC in the External Sync mode.

In the internal synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{\text{RxC}}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

In the Z-80 SIO/2 bonding option, $\overline{\text{SYNCB}}$ is omitted.

TxCA, TxCB. *Transmitter Clocks* (inputs). $\overline{\text{TxC}}$ changes from the falling edge of $\overline{\text{TxC}}$. In asynchronous modes, the Transmitter Clocks may be 1, 16, 32 or 64 times the data rate; however, the clock multiplier for the transmitter and the receiver must be the same. The Transmit Clock inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements (no noise level margin is specified). Transmitter Clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation.

In the Z-80 SIO/0 bonding option, $\overline{\text{TxCB}}$ is bonded together with $\overline{\text{RxCB}}$.

TxDA, TxDB. *Transmit Data* (outputs, active High). Serial data at TTL levels.

W/RDYA, W/RDTB. *Wait/Ready A, Wait/Ready B* (outputs, open drain, when programmed for Wait function; driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the SIO data rate. The reset state is open drain.

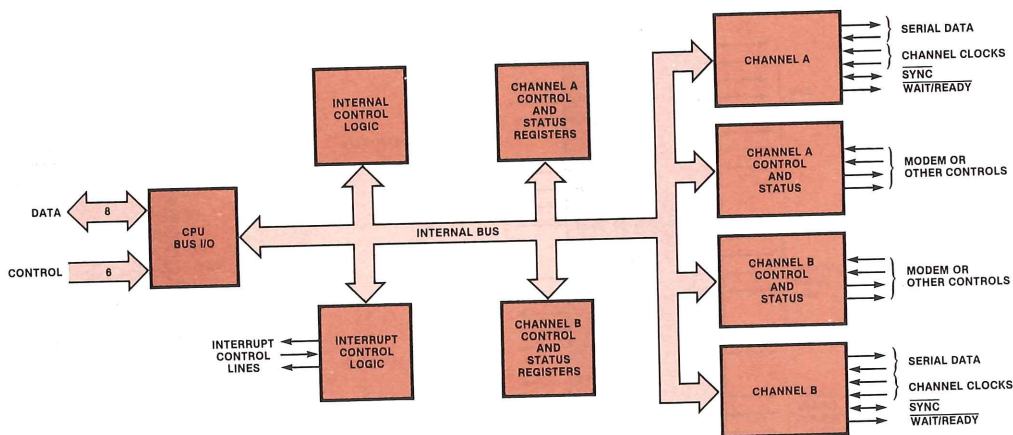


Figure 7. Block Diagram

Functional Description

The functional capabilities of the Z-80 SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data in a wide variety of data-communication protocols; as a Z-80 family peripheral, it interacts with the Z-80 CPU and other peripheral circuits, sharing the data, address and control buses, as well as being a part of the Z-80 interrupt structure. As a peripheral to other microprocessors,

the SIO offers valuable features such as non-vectored interrupts, polling and simple handshake capability.

Figure 8 illustrates the conventional devices that the SIO replaces.

The first part of the following discussion covers SIO data-communication capabilities; the second part describes interactions between the CPU and the SIO.

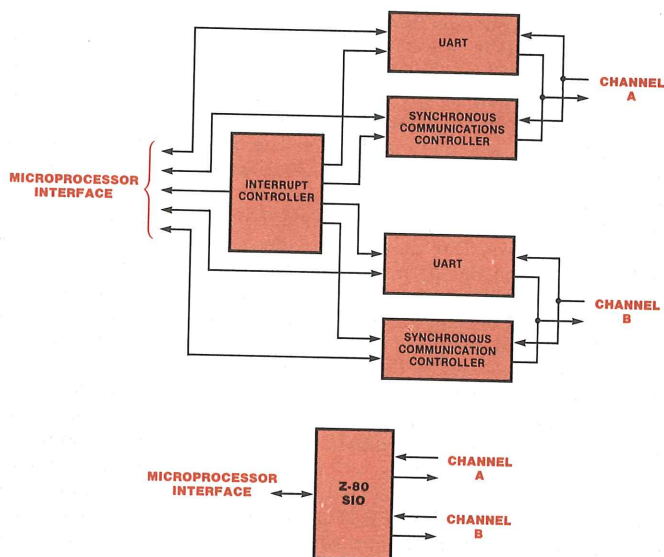


Figure 8. Conventional Devices Replaced by the Z-80 SIO

Data Communication Capabilities

The SIO provides two independent full-duplex channels that can be programmed for use in any common asynchronous or synchronous data-communication protocol. Figure 9 illustrates some of these protocols. The following is a short description of them. A more detailed explanation of these modes can be found in the *Z-80 SIO Technical Manual*.

Asynchronous Modes. Transmission and reception can be done independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 5). If the Low does not persist—as in the case of a transient—the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored

interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The SIO does not require symmetric transmit and receive clock signals—a feature that allows it to be used with a Z-80 CTC or many other clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the receive and transmit clock inputs.

In asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The SIO supports both byte-oriented and bit-oriented synchronous communication.

Synchronous byte-oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16-bit sync pattern (Bisync), or with an external sync signal. Leading sync

characters can be removed without interrupting the CPU.

Five-, six- or seven-bit sync characters are detected with 8- or 16-bit patterns in the SIO by overlapping the larger pattern across multiple in-coming sync characters, as shown in Figure 10.

CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. In all non-SDLC modes, the CRC generator is initialized to 0's; in SDLC modes, it is initialized to 1's. The SIO can be used for interfacing to peripherals such as hard-sectored floppy disk, but it cannot generate or check CRC for IBM-compatible soft-sectored disks. The SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high-speed transmissions under DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length.

The SIO supports synchronous bit-oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message the SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. If a transmit

underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. One to eight bits per character can be sent, which allows reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame in SDLC or HDLC, and provides a synchronization signal on the SYNC pin; an interrupt can also be programmed. The receiver can be programmed to search for frames addressed by a single byte to only a specified user-selected address or to a global broadcast address. In this mode, frames that do not match either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For transmitting data, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers.

The SIO can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SIO then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received.

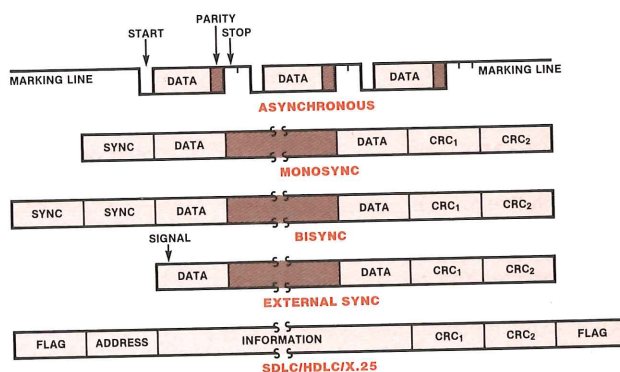


Figure 9. Some Z-80 SIO Protocols

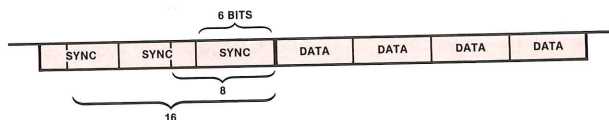


Figure 10.

I/O Interface Capabilities

The SIO offers the choice of polling, interrupt (vectored or non-vectored) and block-transfer modes to transfer data, status and control information to and from the CPU. The block-transfer mode can also be implemented under DMA control.

Polling. Two status registers are updated at appropriate times for each function being performed (for example, CRC error-status valid at the end of a message). When the CPU is operated in a polling fashion, one of the SIO's two status registers is used to indicate whether the SIO has some data or needs some data. Depending on the contents of this register, the CPU will either write data, read data, or just go on. Two bits in the register indicate that a data transfer is needed. In addition, error and other conditions are indicated. The second status register (special receive conditions) does not have to be read in a polling sequence, until a character has been received. All interrupt modes are disabled when operating the device in a polled environment.

Interrupts. The SIO has an elaborate interrupt scheme to provide fast interrupt service in real-time applications. A control register and a status register in Channel B contain the interrupt vector. When programmed to do so, the SIO can modify three bits of the interrupt vector in the status register so that it points directly to one of eight interrupt service routines in memory, thereby servicing conditions in both channels and eliminating most of the needs for a status-analysis routine.

Transmit interrupts, receive interrupts and external/status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control, with Channel A having a higher priority than Channel B, and with receive, transmit and external/status interrupts prioritized in that order within each channel. When the transmit interrupt is enabled, the

CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) The receiver can interrupt the CPU in one of two ways:

- Interrupt on first received character
- Interrupt on all received characters

Interrupt-on-first-received-character is typically used with the block-transfer mode. Interrupt-on-all-received-characters has the option of modifying the interrupt vector in the event of a parity error. Both of these interrupt modes will also interrupt under special receive conditions on a character or message basis (end-of-frame interrupt in SDLC, for example). This means that the special-receive condition can cause an interrupt only if the interrupt-on-first-received-character or interrupt-on-all-received-characters mode is selected. In interrupt-on-first-received-character, an interrupt can occur from special-receive conditions (except parity error) after the first-received-character interrupt (example: receive-overflow interrupt).

The main function of the external/status interrupt is to monitor the signal transitions of the Clear To Send (CTS), Data Carrier Detect (DCD) and Synchronization (SYNC) pins (Figures 1 through 6). In addition, an external/status interrupt is also caused by a CRC-sending condition or by the detection of a break sequence (asynchronous mode) or abort sequence (SDLC mode) in the data stream. The interrupt caused by the break/abort sequence allows the SIO to interrupt when the break/abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the break/abort condition in external logic.

I/O Interface Capabilities (Continued)

In a Z-80 CPU environment (Figure 11), SIO interrupt vectoring is "automatic": the SIO passes its internally-modifiable 8-bit interrupt vector to the CPU, which adds an additional 8 bits from its interrupt-vector (I) register to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. The process entails an indirect transfer of CPU control to the interrupt routine, so that the next instruction executed after an interrupt acknowledge by the CPU is the first instruction of the interrupt routine itself.

CPU/DMA Block Transfer. The SIO's block-transfer mode accommodates both CPU block transfers and DMA controllers (Z-80 DMA or other designs). The block-transfer mode uses the Wait/Ready output signal, which is selected with three bits in an internal control register. The Wait/Ready output signal can be programmed as a WAIT line in the CPU block-transfer mode or as a READY line in the DMA block-transfer mode.

To a DMA controller, the SIO $\overline{\text{READY}}$ output indicates that the SIO is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ output indicates that the SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

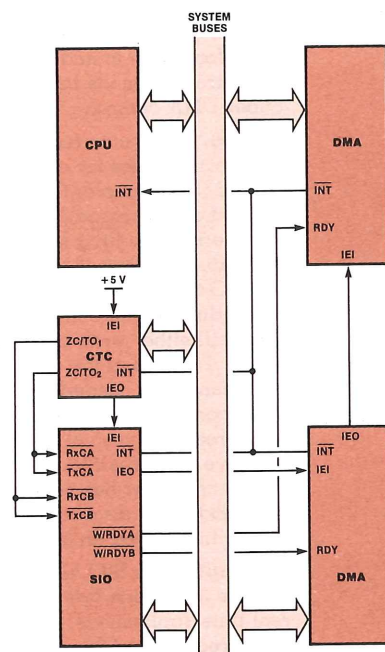


Figure 11. Typical Z-80 Environment

Internal Structure

The internal structure of the device includes a Z-80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains its own set of control and status (write and read) registers, and control and status logic that provides the interface to modems or other external devices.

The registers for each channel are designated as follows:

- WR0-WR7 — Write Registers 0 through 7
- RR0-RR2 — Read Registers 0 through 2

The register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through another 8-bit register (Read Register 2) in Channel B. The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 lists the functions assigned to each read or write register.

Read Register Functions

- | | |
|-----|--|
| RR0 | Transmit/Receive buffer status, interrupt status and external status |
| RR1 | Special Receive Condition status |
| RR2 | Modified interrupt vector (Channel B only) |

Write Register Functions

- | | |
|-----|--|
| WR0 | Register pointers, CRC initialize, initialization commands for the various modes, etc. |
| WR1 | Transmit/Receive interrupt and data transfer mode definition. |
| WR2 | Interrupt vector (Channel B only) |
| WR3 | Receive parameters and control |
| WR4 | Transmit/Receive miscellaneous parameters and modes |
| WR5 | Transmit parameters and controls |
| WR6 | Sync character or SDLC address field |
| WR7 | Sync character or SDLC flag |

Internal Structure (Continued)

The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs, Clear To Send (CTS) and Data Carrier Detect (DCD), are monitored by the external control and status logic under program control. All external control-and-status-logic signals are general-purpose in nature and can be used for functions other than modem control.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 12 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the

CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in asynchronous modes—the character length.

The transmitter has an 8-bit transmit data buffer register that is loaded from the internal data bus, and a 20-bit transmit shift register that can be loaded from the sync-character buffers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxDA).

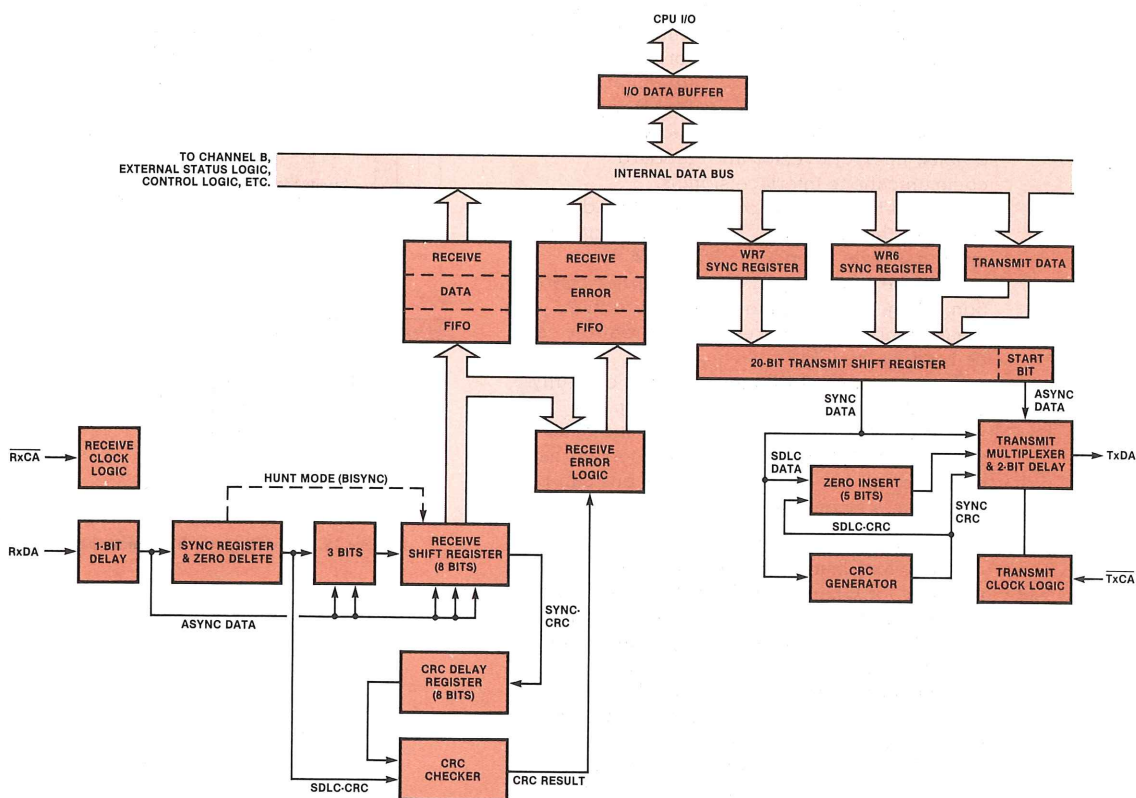


Figure 12. Transmit and Receive Data Path (Channel A)

Programming The system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected mode. For example, the asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first; then the interrupt mode; and finally, receiver or transmitter enable.

Both channels contain registers that must be programmed via the system program prior to operation. The channel-select input ($B/\bar{A}$) and the control/data input ($C/\bar{D}$) are the command-structure addressing controls, and are normally controlled by the CPU address bus. Figures 15 and 16 illustrate the timing relationships for programming the write registers and transferring data and status.

Read Registers. The SIO contains three read registers for Channel B and two read registers for Channel A (RR0-RR2 in Figure 13) that can be read to obtain the status information; RR2 contains the internally-modifiable interrupt vector and is only in the Channel B register set. The status information includes error conditions, interrupt vector and standard communications-interface signals.

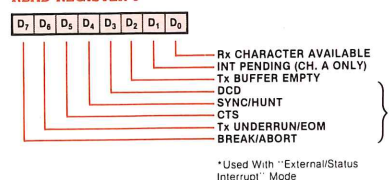
To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing a read instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

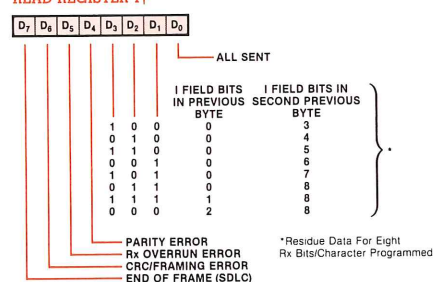
Write Registers. The SIO contains eight write registers for Channel B and seven write registers for Channel A (WR0-WR7 in Figure 14) that are programmed separately to configure the functional personality of the channels; WR2 contains the interrupt vector for both channels and is only in the Channel B register set. With the exception of WR0, programming the write registers requires two bytes. The first byte is to WR0 and contains three bits (D_0 - D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the SIO.

WR0 is a special case in that all of the basic commands can be written to it with a single byte. Reset (internal or external) initializes the pointer bits D_0 - D_2 to point to WR0. This implies that a channel reset must not be combined with the pointing to any register.

READ REGISTER 0

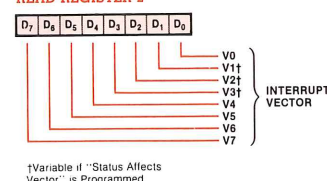


READ REGISTER 1†



†Used With Special Receive Condition Mode

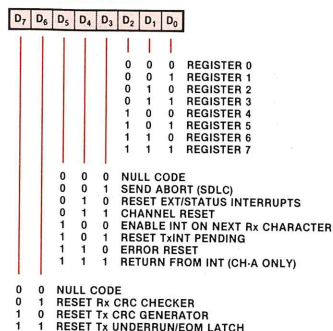
READ REGISTER 2



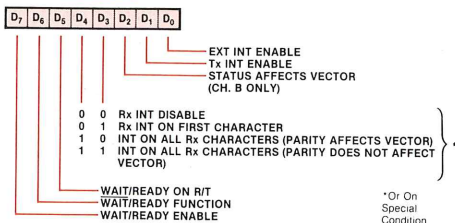
†Variable if "Status Affects Vector" is Programmed

Figure 13. Read Register Bit Functions

WRITE REGISTER 0



WRITE REGISTER 1



The SIO must have the same clock as the CPU (same phase and frequency relationship, not necessarily the same driver).

Read Cycle. The timing signals generated by a Z-80 CPU input instruction to read a data or status byte from the SIO are illustrated in Figure 15.

Write Cycle. Figure 16 illustrates the timing and data signals generated by a Z-80 CPU output instruction to write a data or control byte into the SIO.

Interrupt-Acknowledge Cycle. After receiving an interrupt-request signal from an SIO (INT pulled Low), the Z-80 CPU sends an interrupt-acknowledge sequence ($\overline{M1}$ Low, and $\overline{IORQ}$ Low a few cycles later) as in Figure 17.

The SIO contains an internal daisy-chained interrupt structure for prioritizing nested interrupts for the various functions of its two channels, and this structure can be used within an external user-defined daisy chain that prioritizes several peripheral circuits.

The IEI of the highest-priority device is terminated High. A device that has an interrupt pending or under service forces its IEO Low. For devices with no interrupt pending or under service, $IEO = IEI$.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{M1}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its

internal interrupt-under-service latch.

Return From Interrupt Cycle. Figure 18 illustrates the return from interrupt cycle. Normally, the Z-80 CPU issues a RETI (Return From Interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch in the SIO to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy-chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever "ED" is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is "4D," the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt-acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to the *Z-80 CPU Product Specification*.

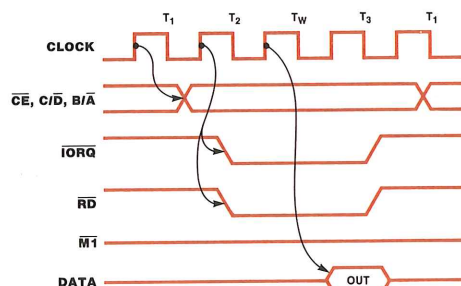


Figure 15. Read Cycle

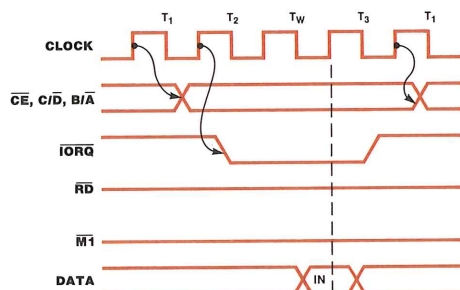


Figure 16. Write Cycle

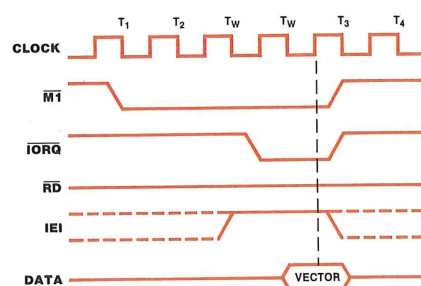


Figure 17. Interrupt Acknowledge Cycle

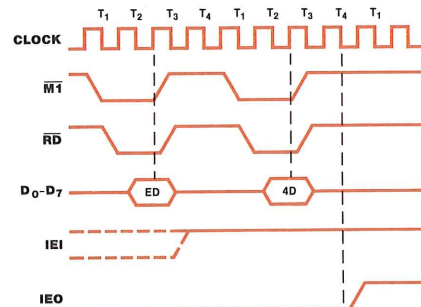


Figure 18. Return from Interrupt Cycle

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND -0.3 V to +7.0 V
 Operating Ambient Temperature As Specified in Ordering Information
 Storage Temperature -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

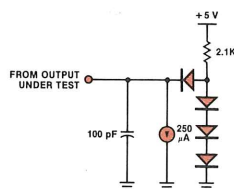
Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\text{ V} \leq V_{CC} \leq +5.25\text{ V}$
- $GND = 0\text{ V}$
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 100 pF max. Timing references between two

output signals assume a load difference of 50 pF max.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	$V_{CC}-0.6$	+5.5	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	$I_{OL} = 2.0\text{ mA}$
V_{OH}	Output High Voltage	+2.4		V	$I_{OH} = -250\text{ μA}$
I_{LI}	Input Leakage Current	-10	+10	μA	$0 < V_{IN} < V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	-10	+10	μA	$0 < V_{IN} < V_{CC}$
$I_{L(SY)}$	$\overline{SYNC}$ Pin Leakage Current	-40	+10	μA	$0 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		100	mA	

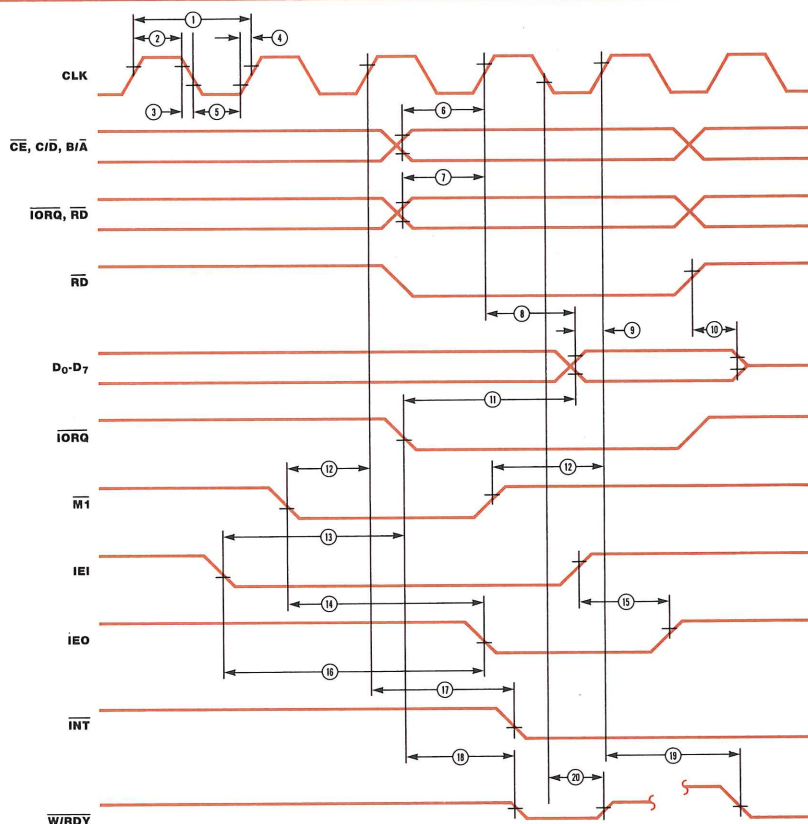
Over specified temperature and voltage range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance	40		pF	Unmeasured
C_{IN}	Input Capacitance		5	pF	pins returned
C_{OUT}	Output Capacitance		10	pF	to ground

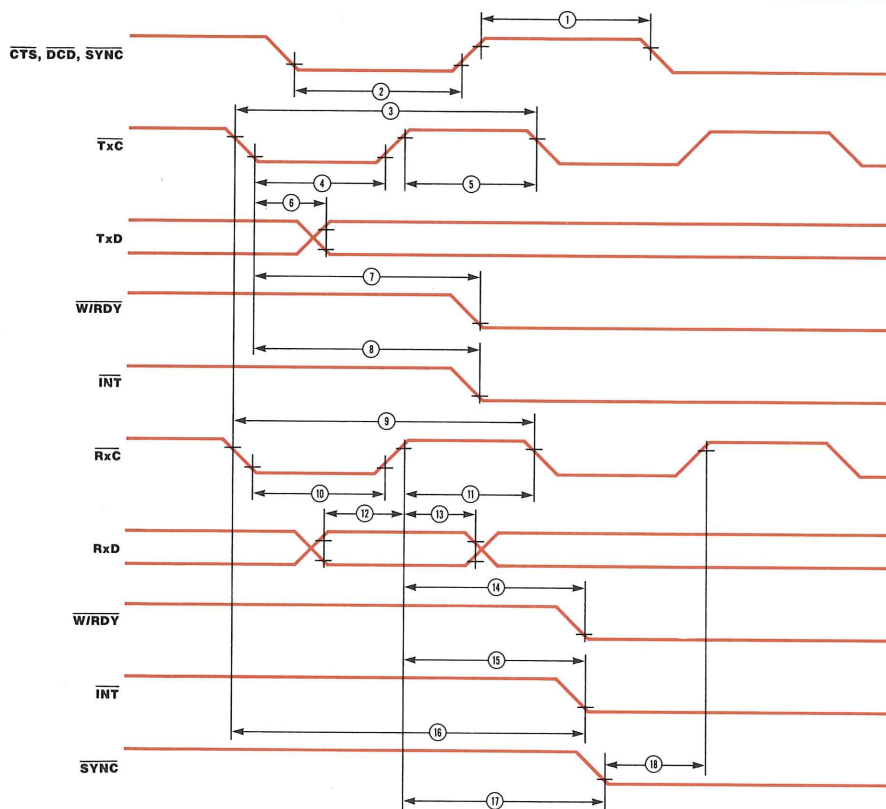
Over specified temperature range; $f = 1\text{ MHz}$

AC Electrical Character- istics



Number	Symbol	Parameter	Z-80 SIO		Z-80A SIO		Unit
			Min	Max	Min	Max	
1	TcC	Clock Cycle Time	400	4000	250	4000	ns
2	TwCh	Clock Width (High)	170	2000	105	2000	ns
3	TfC	Clock Fall Time		30		30	ns
4	TrC	Clock Rise Time		30		30	ns
5	TwCl	Clock Width (Low)	170	2000	105	2000	ns
6	TsAD(C)	$\overline{CE}$, $C/\overline{D}$, $B/\overline{A}$ to Clock $\uparrow$ Setup Time	160		145		ns
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock $\uparrow$ Setup Time	240		115		ns
8	TdC(DO)	Clock $\uparrow$ to Data Out Delay		240		220	ns
9	TsDI(C)	Data In to Clock $\uparrow$ Setup (Write or $\overline{M1}$ Cycle)	50		50		ns
10	TdRD(DOz)	$\overline{RD}$ $\uparrow$ to Data Out Float Delay		230		110	ns
11	TdIO(DOI)	$\overline{IORQ}$ $\downarrow$ to Data Out Delay (INTA Cycle)		340		160	ns
12	TsM1(C)	$\overline{M1}$ to Clock $\uparrow$ Setup Time	210		90		ns
13	TsIEI(IO)	IEI to $\overline{IORQ}$ $\downarrow$ Setup Time (INTA Cycle)	200		140		ns
14	TdM1(IEO)	$\overline{M1}$ $\downarrow$ to IEO $\downarrow$ Delay (interrupt before $\overline{M1}$)	300		190	ns	
15	TdIEI(IEOr)	IEI $\downarrow$ to IEO $\downarrow$ Delay (after ED decode)		150		100	ns
16	TdIEI(IEOf)	IEI $\downarrow$ to IEO $\downarrow$ Delay		150		100	ns
17	TdC(INT)	Clock $\uparrow$ to $\overline{INT}$ $\downarrow$ Delay		200		200	ns
18	TdIO(W/RWf)	$\overline{IORQ}$ $\downarrow$ or $\overline{CE}$ $\downarrow$ to $\overline{W/RDY}$ $\downarrow$ Delay (Wait Mode)		300		210	ns
19	TdC(W/RR)	Clock $\uparrow$ to $\overline{W/RDY}$ $\downarrow$ Delay (Ready Mode)		120		120	ns
20	TdC(W/RWz)	Clock $\downarrow$ to $\overline{W/RDY}$ Float Delay (Wait Mode)		150		130	ns
21	Th	Any unspecified Hold when Setup is specified	0		0		ns

AC
Electrical
Character-
istics
(Continued)



Number	Symbol	Parameter	Z-80 SIO Min	Z-80 SIO Max	Z-80A SIO Min	Z-80A SIO Max	Unit
1	T_{wPh}	Pulse Width (High)	200		200		ns
2	T_{wPl}	Pulse Width (Low)	200		200		ns
3	$T_{cTx C}$	$\overline{Tx C}$ Cycle Time	400	∞	400	∞	ns
4	$T_{wTx Cl}$	$\overline{Tx C}$ Width (Low)	180	∞	180	∞	ns
5	$T_{wTx Ch}$	$\overline{Tx C}$ Width (High)	180	∞	180	∞	ns
6	$T_{dTx C(TxD)}$	$\overline{Tx C} \downarrow$ to $\overline{Tx D}$ Delay (x1 Mode)		400		300	ns
7	$T_{dTx C(W/RRf)}$	$\overline{Tx C} \downarrow$ to $\overline{W/RDY} \downarrow$ Delay (Ready Mode)	5	9	5	9	Clk Periods*
8	$T_{dTx C(INT)}$	$\overline{Tx C} \downarrow$ to $\overline{INT} \downarrow$ Delay	5	9	5	9	Clk Periods*
9	$T_{cRx C}$	$\overline{Rx C}$ Cycle Time	400	∞	400	∞	ns
10	$T_{wRx Cl}$	$\overline{Rx C}$ Width (Low)	180	∞	180	∞	ns
11	$T_{wRx Ch}$	$\overline{Rx C}$ Width (High)	180	∞	180	∞	ns
12	$T_{sRx D(RxC)}$	$\overline{Rx D}$ to $\overline{Rx C} \uparrow$ Setup Time (x1 Mode)	0		0		ns
13	$T_{hRx D(RxC)}$	$\overline{Rx C} \uparrow$ to $\overline{Rx D}$ Hold time (x1 Mode)	140		140		ns
14	$T_{dRx C(W/RRf)}$	$\overline{Rx C} \uparrow$ to $\overline{W/RDY} \downarrow$ Delay (Ready Mode)	10	13	10	13	Clk Periods*
15	$T_{dRx C(INT)}$	$\overline{Rx C} \uparrow$ to $\overline{INT} \downarrow$ Delay	10	13	10	13	Clk Periods*
16	$T_{dTx C(INT)}$	$\overline{Tx C} \downarrow$ to $\overline{INT} \downarrow$ Delay	5	9	5	9	Clk Periods*
17	$T_{dRx C(SYNC)}$	$\overline{Rx C} \uparrow$ to $\overline{SYNC} \downarrow$ Delay (Output Modes)	4	7	4	7	Clk Periods*
18	$T_{sSYNC(RxC)}$	$\overline{SYNC} \downarrow$ to $\overline{Rx C} \uparrow$ Setup (External Sync Modes)	-100		-100		ns

In all modes, the System Clock rate must be at least five times the maximum data rate.
RESET must be active a minimum of one complete Clock Cycle.
*System Clock

Z-80[®] SIO/9 Serial Input/Output Controller



NEW
1980

Product Specification

1

February 1980

Features

- One full-duplex channel, with separate control and status lines for a modem or other device.
- Data rates of 0 to 500K bits/second in the x1 clock mode with a 2.5 MHz clock (Z-80 SIO), or 0 to 800K bits/second with a 4.0 MHz clock (Z-80A SIO).
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7 or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity, overrun and framing error detection.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7 or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection and flag insertion.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

General Description

The Z-80 SIO/9 Serial Input/Output Controller is a single-channel data communication interface with extraordinary versatility and capability. Functionally this device is identical to the Z-80 SIO, except that it operates in one channel only. Its basic functions as a serial-to-

parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions

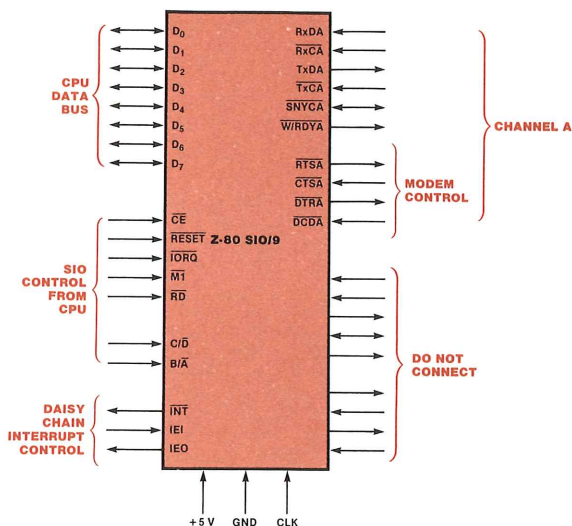


Figure 1. Z-80 SIO/9 Pin Functions

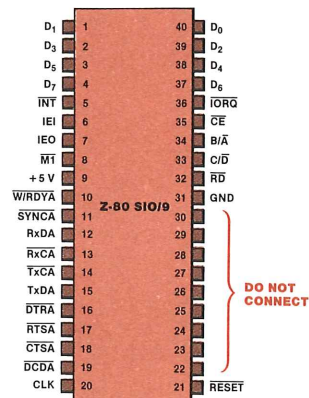


Figure 2. Z-80 SIO/9 Pin Assignments

**General
Description**
(Continued)

traditionally done by UARTs, USARTs and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU and DMA control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast (or slow) peripheral devices. While designed primarily as a member of the Z-80 family, its versatility makes it well-suited to many other CPUs.

The Z-80 SIO/9 is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V

power supply and standard Z-80 family single-phase clock.

Refer to the *Z-80 SIO Product Specification* and the *Z-80 SIO Technical Manual* for detailed functional and electrical descriptions. All functional and electrical descriptions in these publications are applicable to the Z-80 SIO/9, except that Channel B cannot be used for data input or output and pins 22 through 30 must not be connected.

Write Register 2 (interrupt vector) and the Status Affects Vector bit in Write Register 1 are, however, still programmed by selecting Channel B with the B/ $\bar{A}$ input. All other bits in Write Register 1 or Channel B must be programmed to 0.

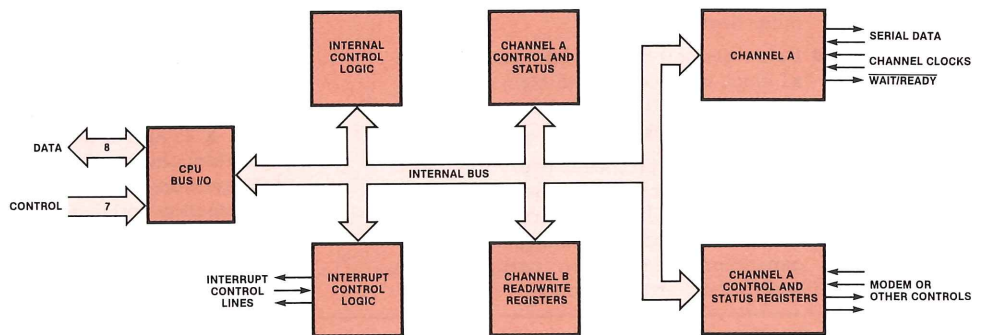


Figure 3. Block Diagram

Z-80[®] DART Dual Asynchronous Receiver/Transmitter



NEW
1980

Product Specification

1

February 1980

Features

- Two independent full-duplex channels with separate modem controls. Modem status can be monitored.
- Receiver data registers are quadruply buffered; the transmitter is doubly buffered.
- Interrupt features include a programmable interrupt vector, a "status affects vector" mode for fast interrupt processing, and the standard Z-80 peripheral daisy-chain interrupt structure that provides automatic interrupt vectoring with no external logic.
- In x1 clock mode, data rates are 0 to 500K bits/second with a 2.5 MHz clock, or 0 to 800K bits/second with a 4.0 MHz clock.
- Programmable options include 1, 1½ or 2 stop bits; even, odd or no parity; and x1, x16, x32 and x64 clock modes.
- Break generation and detection as well as parity-, overrun- and framing-error detection are available.

Description

The Z-80 DART (Dual-Channel Asynchronous Receiver/Transmitter) is a dual-channel multi-function peripheral component that satisfies a wide variety of asynchronous serial data communications requirements in micro-computer systems. The Z-80 DART is used as a serial-to-parallel, parallel-to-serial converter/controller in asynchronous applications. In addition, the device also provides modem controls for both channels. In applications where

modem controls are not needed, these lines can be used for general-purpose I/O.

Zilog also offers the Z-80 SIO, a more versatile device that provides synchronous (Bisync, HDLC and SDLC) as well as asynchronous operation.

The Z-80 DART is fabricated with n-channel silicon-gate depletion-load technology, and is packaged in a 40-pin plastic or ceramic DIP.

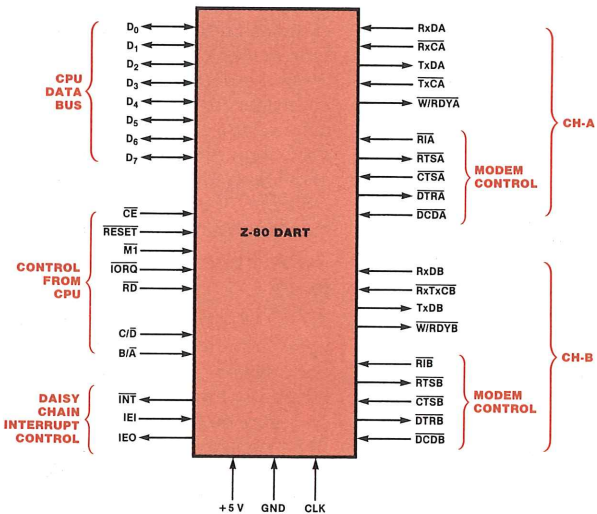


Figure 1. Z-80 DART Pin Functions

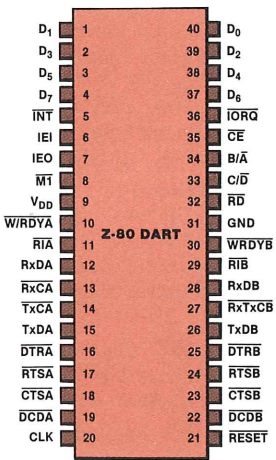


Figure 2. Pin Assignments

Pin	Description
-----	-------------

B/ $\overline{A}$. *Channel A Or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the Z-80 DART.

C/ $\overline{D}$. *Control Or Data Select* (input, High selects Control). This input specifies the type of information (control or data) transferred on the data bus between the CPU and the Z-80 DART.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low at this input enables the Z-80 DART to accept command or data input from the CPU during a write cycle, or to transmit data to the CPU during a read cycle.

CLK. *System Clock* (input). The Z-80 DART uses the standard Z-80 single-phase system clock to synchronize internal signals.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals.

D_0 - D_7 . *System Data Bus* (bidirectional, 3-state) transfers data and commands between the CPU and the Z-80 DART.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the Z-80 DART is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered.

$\overline{DTRA}$, $\overline{DTRB}$. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be programmed as general-purpose outputs.

IEI. *Interrupt Enable In* (input, active High) is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this Z-80 DART. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When the Z-80 DART is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{M1}$. *Machine Cycle One* (input from Z-80 CPU, active Low). When $\overline{M1}$ and $\overline{RD}$ are both active, the Z-80 CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the Z-80 DART accepts $\overline{M1}$ and $\overline{IORQ}$

as an interrupt acknowledge if the Z-80 DART is the highest priority device that has interrupted the Z-80 CPU.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with $B/\overline{A}$, $C/\overline{D}$, $\overline{CE}$ and $\overline{RD}$ to transfer commands and data between the CPU and the Z-80 DART. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are all active, the channel selected by $B/\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active, but $\overline{RD}$ is inactive, the channel selected by $B/\overline{A}$ is written to by the CPU with either data or control information as specified by $C/\overline{D}$.

$RxC\overline{A}$, $RxC\overline{B}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of RxC . The Receive Clocks may be 1, 16, 32 or 64 times the data rate.

$\overline{RD}$. *Read Cycle Status*. (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress.

$RxDA$, $RxDB$. *Receive Data* (inputs, active High).

$\overline{RESET}$. *Reset* (input, active Low). Disables both receivers and transmitters, forces $TxDA$ and $TxDB$ marking, forces the modem controls High and disables all interrupts.

$\overline{RIA}$, $\overline{RIB}$. *Ring Indicator* (inputs, Active Low). These inputs are similar to $\overline{CTS}$ and $\overline{DCD}$. The Z-80 DART detects both logic level transitions and interrupts the CPU. When not used in switched-line applications, these inputs can be used as general-purpose inputs.

$\overline{RTSA}$, $\overline{RTSB}$. *Request to Send* (outputs, active Low). When the RTS bit is set, the $\overline{RTS}$ output goes Low. When the RTS bit is reset, the output goes High after the transmitter empties.

$TxC\overline{A}$, $TxC\overline{B}$. *Transmitter Clocks* (inputs). TxD changes on the falling edge of TxC . The Transmitter Clocks may be 1, 16, 32 or 64 times the data rate; however, the clock multiplier for the transmitter and the receiver must be the same. The Transmit Clock inputs are Schmitt-trigger buffered. Both the Receiver and Transmitter Clocks may be driven by the Z-80 CTC Counter Time Circuit for programmable baud rate generation.

$TxDA$, $TxDB$. *Transmit Data* (outputs, active High).

$\overline{W/RDYA}$, $\overline{W/RDYB}$. *Wait/Ready* (outputs, open drain when programmed for Wait function, driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the Z-80 DART data rate. The reset state is open drain.

Functional Description

The functional capabilities of the Z-80 DART can be described from two different points of view: as a data communications device, it transmits and receives serial data, and meets the requirements of asynchronous data communications protocols; as a Z-80 family peripheral, it interacts with the Z-80 CPU and other Z-80 peripheral circuits, and shares the data, address and control buses, as well as being a part of the Z-80 interrupt structure. As a peripheral to other microprocessors, the Z-80 DART offers valuable features such as non-vectored interrupts, polling and simple hand-

shake capability.

The first part of the following functional description introduces Z-80 DART data communications capabilities; the second part describes the interaction between the CPU and the Z-80 DART.

A more detailed explanation of Z-80 DART operation can be found in the *Z-80 SIO Technical Manual* (Document Number 03-3033-01). Because this manual was written for the Z-80 SIO, it contains information about synchronous as well as asynchronous operation.

Communications Capabilities. The Z-80 DART provides two independent full-duplex channels for use as an asynchronous receiver/transmitter. The following is a short description of receiver/transmitter capabilities. For more details, refer to the Asynchronous Mode section of the *Z-80 SIO Technical Manual*. The Z-80 DART offers transmission and reception of five to eight bits per character, plus optional even or odd parity. The transmitter can supply one, one and a half or two stop bits per character and can provide a break output at any time. The receiver break detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the Receive Data input. If the Low does not persist—as in the case of a transient—the character assembly process is not started.

I/O Interface Capabilities. The Z-80 DART offers the choice of Polling, Interrupt (vectored or non-vectored) and Block Transfer modes to transfer data, status and control information to

Framing errors and overrun errors are detected and buffered together with the character on which they occurred. Vectored interrupts allow fast servicing of interrupting conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The Z-80 DART does not require symmetric Transmit and Receive Clock signals—a feature that allows it to be used with a Z-80 CTC or any other clock source. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the Receive and Transmit Clock inputs. When using Channel B, the bit rates for transmit and receive operations must be the same because RxC and TxC are bonded together ($RxTxCB$).

and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

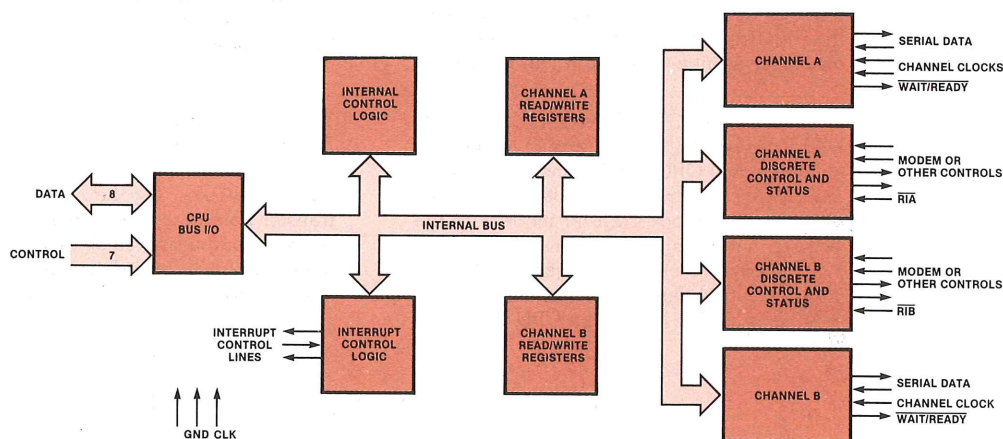


Figure 3. Block Diagram

POLLING. There are no interrupts in the Polled mode. Status registers RR0 and RR1 are updated at appropriate times for each function being performed. All the interrupt modes of the Z-80 DART must be disabled to operate the device in a polled environment.

While in its Polling sequence, the CPU examines the status contained in RR0 for each channel; the RR0 status bits serve as an acknowledge to the Poll inquiry. The two RR0

status bits D_0 and D_2 indicate that a data transfer is needed. The status also indicates Error or other special status conditions (see "Z-80 DART Programming"). The Special Receive Condition status contained in RR1 does not have to be read in a Polling sequence because the status bits in RR1 are accompanied by a Receive Character Available status in RR0.

INTERRUPTS. The Z-80 DART offers an elaborate interrupt scheme that provides fast interrupt response in real-time applications. As a member of the Z-80 family, the Z-80 DART can be daisy-chained along with other Z-80 peripherals for peripheral interrupt-priority resolution. In addition, the internal interrupts of the Z-80 DART are nested to prioritize the various interrupts generated by Channels A and B. Channel B registers WR2 and RR2 contain the interrupt vector that points to an interrupt service routine in the memory. To eliminate the necessity of writing a status analysis routine, the Z-80 DART can modify the interrupt vector in RR2 so it points directly to one of eight interrupt service routines. This is done under program control by setting a program bit ($WR1, D_2$) in Channel B called "Status Affects Vector." When this bit is set, the interrupt vector in RR2 is modified according to the assigned priority of the various interrupting conditions.

Transmit interrupts, Receive interrupts and External/Status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control with Channel A having a higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted by the transmit buffer *becoming* empty. (This implies that the transmitter must have had a data character written into it so it can become

empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on the first received character
- Interrupt on all received characters
- Interrupt on a Special Receive condition

Interrupt On First Character is typically used with the Block Transfer mode. Interrupt On All Receive Characters can optionally modify the interrupt vector in the event of a parity error. The Special Receive Condition interrupt can occur on a character basis. The Special Receive condition can cause an interrupt only if the Interrupt On First Receive Character or Interrupt On All Receive Characters mode is selected. In Interrupt On First Receive Character, an interrupt can occur from Special Receive conditions (except Parity Error) after the first receive character interrupt (example: Receive Overrun interrupt).

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$ and $\overline{RI}$ pins; however, an External/Status interrupt is also caused by the detection of a Break sequence in the data stream. The interrupt caused by the Break sequence has a special feature that allows the Z-80 DART to interrupt when the Break sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Break condition.

CPU/DMA BLOCK TRANSFER. The Z-80 DART provides a Block Transfer mode to accommodate CPU block transfer functions and DMA block transfers (Z-80 DMA or other designs). The Block Transfer mode uses the $\overline{WRDY}$ output in conjunction with the Wait/Ready bits of Write Register 1. The $\overline{WRDY}$ output can be defined under software control as a Wait line in the CPU Block

Transfer mode or as a Ready line in the DMA Block Transfer mode.

To a DMA controller, the Z-80 DART Ready output indicates that the Z-80 DART is ready to transfer data to or from memory. To the CPU, the Wait output indicates that the Z-80 DART is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

Internal Architecture

The device internal structure includes a Z-80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains read and write registers, and discrete control and status logic that provides the interface to modems or other external devices.

The read and write register group includes five 8-bit control registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through Read Register 2 in Channel B. The registers for both channels are designated as follows:

WR0-WR5 — Write Registers 0 through 5

RR0-RR2 — Read Registers 0 through 2

The bit assignment and functional grouping of each register is configured to simplify and

organize the programming process.

The logic for both channels provides formats, bit synchronization and validation for data transferred to and from the channel interface. The modem control inputs Clear to Send (CTS), Data Carrier Detect (DCD) and Ring Indicator (RI) are monitored by the control logic under program control. All the modem control signals are general purpose in nature and can be used for functions other than modem control.

For automatic interrupt vectoring, the interrupt control logic determines which channel and which device within the channel has the highest priority. Priority is fixed with Channel A assigned a higher priority than Channel B; Receive, Transmit and External/Status interrupts are prioritized in that order within each channel.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 4 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to

service a Receive Character Available interrupt in a high-speed data transfer.

The transmitter has an 8-bit transmit data register that is loaded from the internal data bus, and a 9-bit transmit shift register that is loaded from the transmit data register.

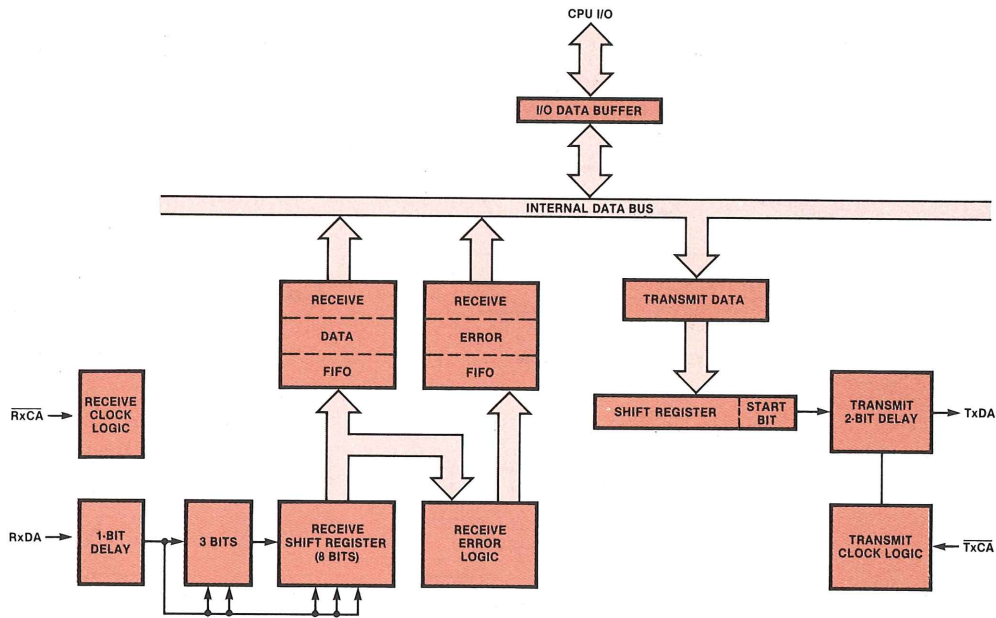


Figure 4. Data Path

Read Cycle. The timing signals generated by a Z-80 CPU input instruction to read a Data or

Status byte from the Z-80 DART are illustrated in Figure 5a.

Write Cycle. Figure 5b illustrates the timing and data signals generated by a Z-80 CPU out-

put instruction to write a Data or Control byte into the Z-80 DART.

Interrupt Acknowledge Cycle. After receiving an Interrupt Request signal (INT pulled Low), the Z-80 CPU sends an Interrupt Acknowledge signal ($\overline{MI}$ and $\overline{IORQ}$ both Low). The daisy-chained interrupt circuits determine the highest priority interrupt requestor. The IEI of the highest priority peripheral is terminated High. For any peripheral that has no interrupt pending or under service, $IEO = IEI$. Any peripheral that does have an interrupt pending or under service forces its IEO Low.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{MI}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Refer to the *Z-80 SIO Technical Manual* for additional details on the interrupt daisy chain and interrupt nesting.

Return From Interrupt Cycle. Normally, the Z-80 CPU issues an RETI (Return From Interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch to terminate the interrupt that has just been processed.

When used with other CPUs, the Z-80 DART allows the user to return from the interrupt cycle with a special command called "Return From Interrupt" in Write Register 0 of Channel A. This command is interpreted by the Z-80 DART in exactly the same way it would interpret an RETI command on the data bus.

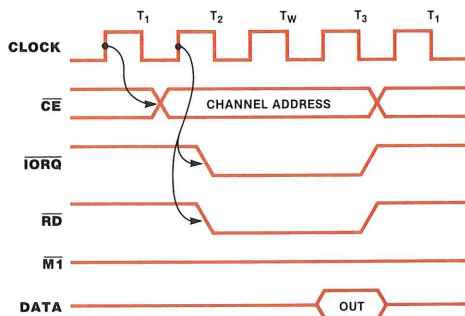


Figure 5a. Read Cycle

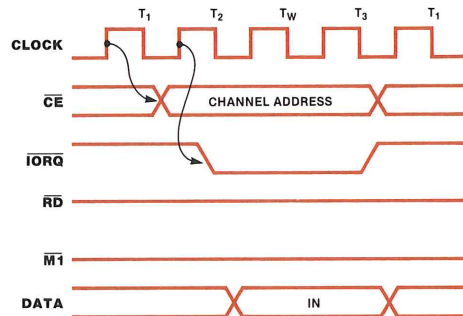


Figure 5b. Write Cycle

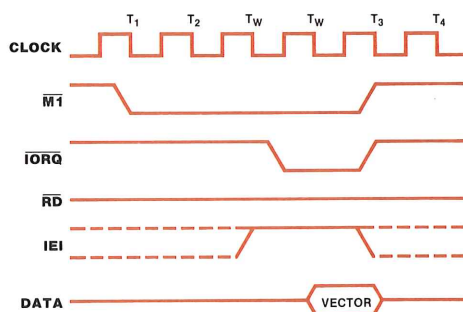


Figure 5c. Interrupt Acknowledge Cycle

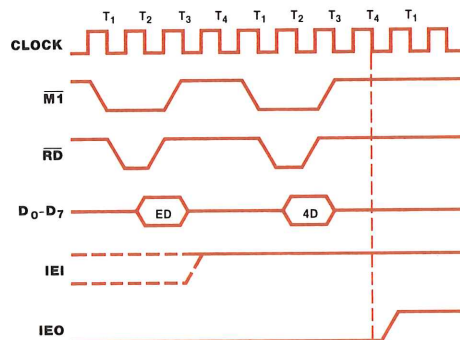


Figure 5d. Return from Interrupt Cycle

**Z-80 DART
Programming**

To program the Z-80 DART, the system program first issues a series of commands that initialize the basic mode and then other commands that qualify conditions within the selected mode. For example, the character length, clock rate, number of stop bits, even or odd parity are first set, then the Interrupt mode and, finally, receiver or transmitter enable.

Both channels contain command registers that must be programmed via the system program prior to operation. The Channel Select input (B/ $\overline{A}$) and the Control/Data input (C/ $\overline{D}$) are the command structure addressing controls, and are normally controlled by the CPU address bus.

Write Registers. The Z-80 DART contains six registers (WR0-WR5) in each channel that are programmed separately by the system program to configure the functional personality of the channels (Figure 4). With the exception of WR0, programming the write registers requires two bytes. The first byte contains three bits (D_0 - D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the Z-80 DART.

WR0 is a special case in that all the basic commands (CMD₀-CMD₂) can be accessed with a single byte. Reset (internal or external) initializes the pointer bits D_0 - D_2 to point to WR0. This means that a register cannot be

pointed to in the same operation as a channel reset.

Write Register Functions

WR0	Register pointers, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls

Read Registers. The Z-80 DART contains three registers (RR0-RR2) that can be read to obtain the status information for each channel (except for RR2, which applies to Channel B only). The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing an input instruction, the contents of the addressed read register can be read by the CPU.

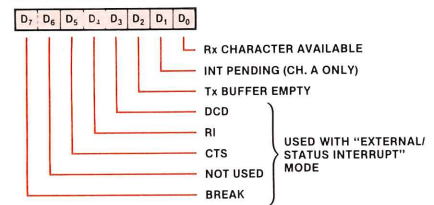
The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Read Register Functions

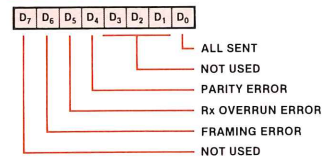
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

Z-80 DART Read and Write Registers

READ REGISTER 0

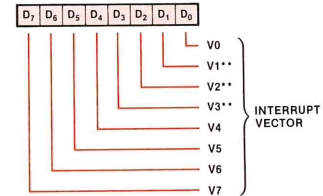


READ REGISTER 1*



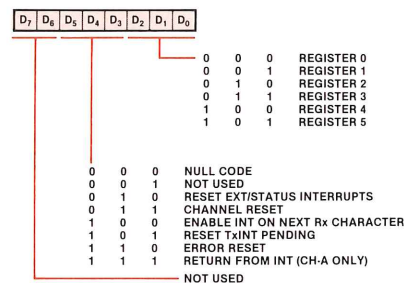
*Used With Special Receive Condition Mode

READ REGISTER 2

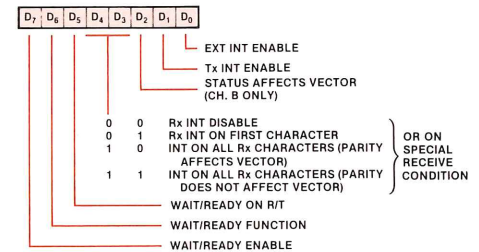


**Variable If "Status Affects Vector" Is Programmed

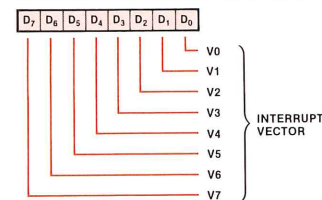
WRITE REGISTER 0



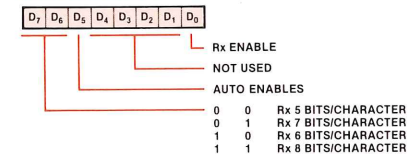
WRITE REGISTER 1



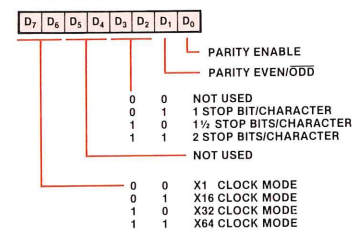
WRITE REGISTER 2 (CHANNEL B ONLY)



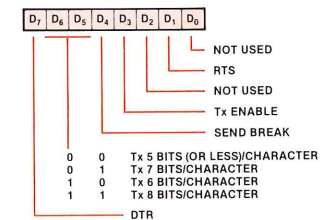
WRITE REGISTER 3



WRITE REGISTER 4



WRITE REGISTER 5



Absolute Maximum Ratings

Voltages on all inputs and outputs
with respect to GND -0.3 V to +7.0 V
Operating Ambient
Temperature As Specified in
Ordering Information
Storage Temperature -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

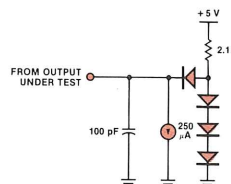
Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\text{ V} \leq V_{CC} \leq +5.25\text{ V}$
- $GND = 0\text{ V}$
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 100 pF max. Timing references between two

output signals assume a load difference of 50 pF max.

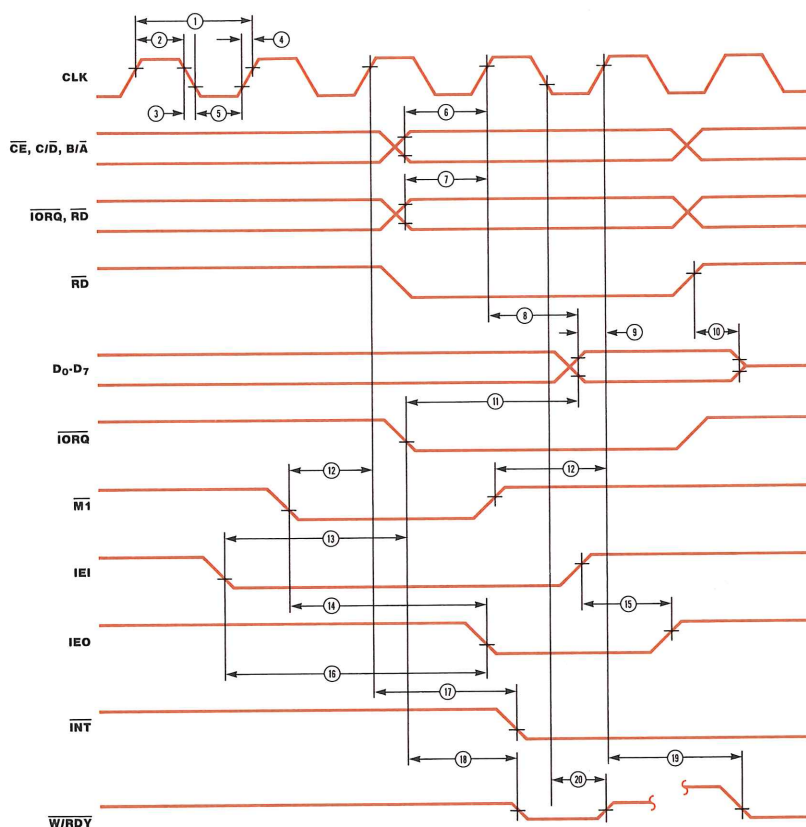


DC Charac- teristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	$V_{CC}-0.6$	+5.5	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	$I_{OL} = 2.0\text{ mA}$
V_{OH}	Output High Voltage	+2.4		V	$I_{OH} = -250\text{ }\mu\text{A}$
I_L	Input/3-State Output Leakage Current	-10	+10	μA	$0.4 < V < 2.4\text{ V}$
$I_{L(RI)}$	RI Pin Leakage Current	-40	+10	μA	$0.4 < V < 2.4\text{ V}$
I_{CC}	Power Supply Current		100	mA	

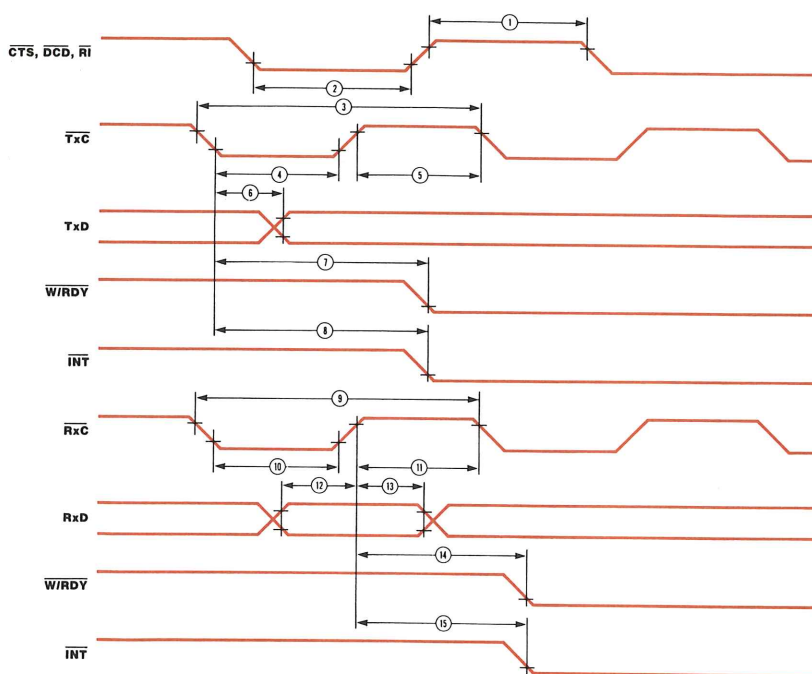
$T_A = 0^\circ\text{C to } 70^\circ\text{C}, V_{CC} = +5\text{V}, \pm 5\%$

AC Electrical Character- istics



Number	Symbol	Parameter	Z-80 DART		Z-80A DART		Unit
			Min	Max	Min	Max	
1	T _c C	Clock Cycle Time	400	4000	250	4000	ns
2	T _w Ch	Clock Width (High)	170	2000	105	2000	ns
3	T _f C	Clock Fall Time		30		30	ns
4	T _r C	Clock Rise Time		30		30	ns
5	T _w C _l	Clock Width (Low)	170	2000	105	2000	ns
6	T _s AD(C)	$\overline{CE}$, $C/\overline{D}$, $B/\overline{A}$ to Clock $\uparrow$ Setup Time	160		145		ns
7	T _s CS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock $\uparrow$ Setup Time	240		115		ns
8	T _d C(DO)	Clock $\uparrow$ to Data Out Delay		240		220	ns
9	T _s DI(C)	Data In to Clock $\uparrow$ Setup Time	50		50		ns
10	T _d RD(DO _z)	$\overline{RD}$ $\uparrow$ to Data Out Float Delay		230		110	ns
11	T _d IO(DOI)	$\overline{IORQ}$ $\uparrow$ to Data Out Delay (INTA Cycle)		340		160	ns
12	T _s M ₁ (C)	$\overline{M1}$ to Clock $\uparrow$ Setup Time	210		90		ns
13	T _s IEI(IO)	IEI to $\overline{IORQ}$ $\downarrow$ Setup Time (INTA Cycle)	200		140		ns
14	T _d M ₁ (IEO)	$\overline{M1}$ $\downarrow$ to IEO $\downarrow$ Delay (interrupt immediately preceding $\overline{M1}$ $\downarrow$)		300		190	ns
15	T _d IEI(IEO _r)	IEI $\uparrow$ to IEO $\downarrow$ Delay (after ED decode)		150		100	ns
16	T _d IEI(IEO _f)	IEI $\downarrow$ to IEO $\downarrow$ Delay		150		100	ns
17	T _d C(INT)	Clock $\uparrow$ to $\overline{INT}$ $\downarrow$ Delay		200		200	ns
18	T _d IO(W/RW _f)	$\overline{IORQ}$ $\downarrow$ or $\overline{CE}$ $\downarrow$ to $\overline{WRDY}$ $\downarrow$ Delay (Wait Mode)		300		210	ns
19	T _d C(W/RR)	Clock $\uparrow$ to $\overline{WRDY}$ $\downarrow$ Delay (Ready Mode)		120		120	ns
20	T _d C(W/RW _z)	Clock $\uparrow$ to $\overline{WRDY}$ Float Delay (Wait Mode)		150		130	ns

AC
Electrical
Character-
istics
(Continued)



1

Number	Symbol	Parameter	Z-80 DART		Z-80A DART		Unit
			Min	Max	Min	Max	
1	TwPh	Pulse Width (High)	200		200		ns
2	TwPl	Pulse Width (Low)	200		200		ns
3	TcTx̄C	Tx̄C Cycle Time	400	∞	400	∞	ns
4	TwTx̄C1	Tx̄C Width (Low)	180	∞	180	∞	ns
5	TwTx̄Ch	Tx̄C Width (High)	180	∞	180	∞	ns
6	TdTx̄C(TxD)	Tx̄C ↓ to TxD Delay		400		300	ns
7	TdTx̄C(W/RRf)	Tx̄C ↓ to W/ RDY ↓ Delay (Ready Mode)	5	9	5	9	Clk Periods
8	TdTx̄C(INT)	Tx̄C ↓ to INT̄ ↓ Delay	5	9	5	9	Clk Periods
9	TcRx̄C	Rx̄C Cycle Time	400	∞	400	∞	ns
10	TwRx̄C1	Rx̄C Width (Low)	180	∞	180	∞	ns
11	TwRx̄Ch	Rx̄C Width (High)	180	∞	180	∞	ns
12	TsRx̄D(RxC)	RxD to Rx̄C ↑ Setup Time (x1 Mode)	0		0		ns
13	ThRx̄D(RxC)	RxD Hold time (x1 Mode)	140		140		ns
14	TdRx̄C(W/RRf)	Rx̄C ↑ to W/ RDY ↓ Delay (Ready Mode)	10	13	10	13	Clk Periods
15	TdRx̄C(INT)	Rx̄C ↑ to INT̄ ↓ Delay	10	13	10	13	Clk Periods

In all modes, the Clock rate must be at least five times the maximum data rate.
RESET must be active a minimum of one complete Clock Cycle.

Z-80[®] DMA Direct Memory Access Controller



**NEW
1980**

Product Specification

1

February 1980

Features

- Transfers, searches and search/transfers in Byte-at-a-Time, Burst or Continuous modes. Cycle length and edge timing can be programmed to match the speed of any port.
- Dual port addresses (source and destination) generated for memory-to-I/O, memory-to-memory, or I/O-to-I/O operations. Addresses may be fixed or automatically incremented/decremented.
- Next-operation loading without disturbing current operations via buffered starting-

address registers. An entire previous sequence can be repeated automatically.

- Extensive programmability of functions. CPU can read complete channel status.
- Standard Z-80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic. Sophisticated, internally modifiable interrupt vectoring.
- Direct interfacing to system buses without external logic.

General Description

The Z-80 DMA (Direct Memory Access) is a powerful and versatile device for controlling and processing transfers of data. Its basic function of managing CPU-independent transfers between two ports is augmented by an array of features that optimize transfer speed and control with little or no external logic in systems using an 8- or 16-bit data bus and a 16-bit address bus.

Transfers can be done between any two ports (source and destination), including memory-to-I/O, memory-to-memory, and I/O-to-I/O. Dual port addresses are automatically generated for each transaction and may be either fixed or incrementing/decrementing. In addition, bit-maskable byte searches can be performed either concurrently with transfers or as an operation in itself.

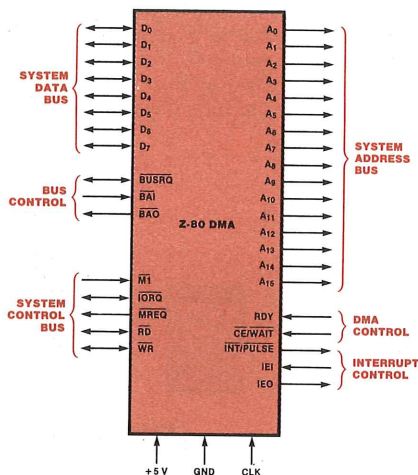


Figure 1. Pin Functions

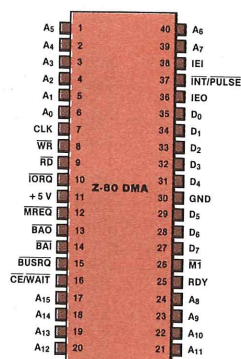


Figure 2. Pin Assignments

General Description (Continued)

The Z-80 DMA contains direct interfacing to and independent control of system buses, as well as sophisticated bus and interrupt controls. Many programmable features, including variable cycle timing and auto-restart, minimize CPU software overhead. They are especially useful in adapting this special-

purpose transfer processor to a broad variety of memory, I/O and CPU environments.

The Z-80 DMA is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V power supply and the standard Z-80 Family single-phase clock.

Functional Description

Classes of Operation. The Z-80 DMA has three basic classes of operation:

- Transfers of data between two ports (memory or I/O peripheral)
- Searches for a particular 8-bit maskable byte at a single port in memory or an I/O peripheral
- Combined transfers with simultaneous search between two ports

Figure 4 illustrates the basic functions served by these classes of operation.

During a transfer, the DMA assumes control of the system address and data buses. Data is read from one addressable port and written to the other addressable port, byte by byte. The ports may be programmed to be either system main memory or peripheral I/O devices. Thus, a block of data may be written from one peripheral to another, from one area of main memory to another, or from a peripheral to main memory and vice versa.

During a search-only operation, data is read from the source port and compared byte by byte with a DMA-internal register containing a programmable match byte. This match byte may optionally be masked so that only certain bits within the match byte are compared. Search rates up to 1.25M bytes per second can be obtained with the 2.5 MHz Z-80 DMA or 2M bytes per second with the 4 MHz Z-80A DMA.

In combined searches and transfers, data is transferred between two ports while simultaneously searching for a bit-maskable byte match.

Data transfers or searches can be programmed to stop or interrupt under various conditions. In addition, CPU-readable status bits can be programmed to reflect the condition.

Modes of Operation. The Z-80 DMA can be programmed to operate in one of three transfer and/or search modes:

- **Byte-at-a-Time:** data operations are performed one byte at a time. Between each byte operation the system buses are released to the CPU. The buses are requested again for each succeeding byte operation.
- **Burst:** data operations continue until a port's Ready line to the DMA goes inactive. The DMA then stops and releases the system buses after completing its current byte operation.
- **Continuous:** data operations continue until the end of the programmed block of data is reached before the system buses are released. If a port's Ready line goes inactive before this occurs, the DMA simply pauses until the Ready line comes active again.

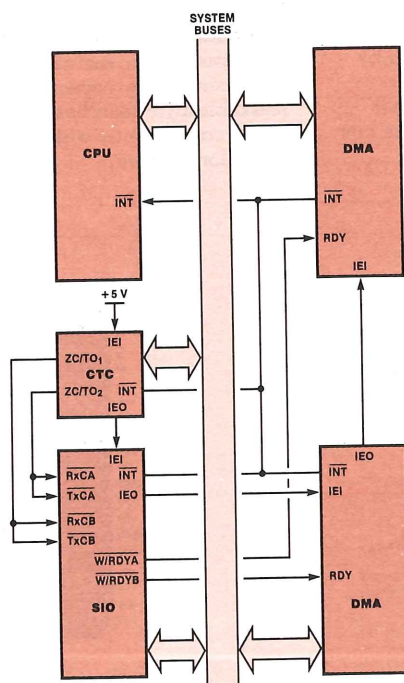
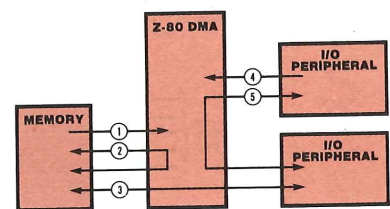


Figure 3. Typical Z-80 Environment



1. Search memory
2. Transfer memory-to-memory (optional search)
3. Transfer memory-to-I/O (optional search)
4. Search I/O
5. Transfer I/O-to-I/O (optional search)

Figure 4. Basic Functions of the Z-80 DMA

Functional Description (Continued)

In all modes, once a byte of data is read into the DMA, the operation on the byte will be completed in an orderly fashion, regardless of the state of other signals (including a port's Ready line).

Due to the DMA's high-speed buffered method of reading data, operations on one byte are not completed until the next byte is read in. This means that total transfer or search block lengths must be two or more bytes, and that block lengths programmed into the DMA must be one byte less than the desired block length (count is $N-1$ where N is the block length).

Commands and Status. The Z-80 DMA has several writable control registers and readable status registers available to the CPU. Control bytes can be written to the DMA whenever the DMA is not controlling the system buses, but the act of writing a control byte to the DMA disables the DMA until it is again enabled by a specific command. Status bytes can also be read at any such time, but writing the Read Status Byte command or the Initiate Read Sequence command disables the DMA.

Control bytes to the DMA include those which effect immediate command actions such as enable, disable, reset, load starting-address buffers, continue, clear counters, clear status bits and the like. In addition, many mode-setting control bytes can be written, including mode and class of operation, port configuration, starting addresses, block length, address counting rule, match and match-mask byte, interrupt conditions, interrupt vector, status-affects-vector condition, pulse counting, auto restart, Ready-line and Wait-line rules, and read mask.

Readable status registers include a general status byte reflecting Ready-line, end-of-block, byte-match and interrupt conditions, as well as 2-byte registers for the current byte count, Port A address and Port B address.

Variable Cycle. The Z-80 DMA has the unique feature of programmable operation-cycle length. This is valuable in tailoring the DMA to the particular requirements of other system components (fast or slow) and maximizes the data-transfer rate. It also eliminates external logic for signal conditioning.

There are two aspects to the variable cycle feature. First, the entire read and write cycles (periods) associated with the source and destination ports can be independently programmed as 2, 3 or 4 T-cycles long (more if Wait cycles are used), thereby increasing or decreasing the speed with which all DMA signals change (Figure 5).

Second, the four signals in each port specifically associated with transfers of data (I/O Request, Memory Request, Read and Write) can each have its active trailing edge terminated one-half T-cycle early. This adds a further dimension of flexibility and speed, allowing such things as shorter-than-normal Read or Write signals that go inactive before data starts to change.

Address Generation. Two 16-bit addresses are generated by the Z-80 DMA for every transfer operation, one address for the source port and another for the destination port. Each address can be either variable or fixed. Variable addresses can increment or decrement from the programmed starting address. The fixed-address capability eliminates the need for separate enabling wires to I/O ports.

Port addresses are multiplexed onto the system address bus, depending on whether the DMA is reading the source port or writing to the destination port. Two readable address counters (2 bytes each) keep the current address of each port.

Auto Restart. The starting addresses of either port can be reloaded automatically at the end of a block. This option is selected by the Auto Restart control bit. The byte counter is cleared when the addresses are reloaded.

The Auto Restart feature relieves the CPU of software overhead for repetitive operations such as CRT refresh and many others. Moreover, when the CPU has access to the buses during byte-at-a-time or burst transfers, different starting addresses can be written into buffer registers during transfers, causing the Auto Restart to begin at a new location.

Interrupts. The Z-80 DMA can be programmed to interrupt the CPU on four conditions:

- Interrupt on Ready (before requesting bus)
- Interrupt on Match
- Interrupt on End of Block

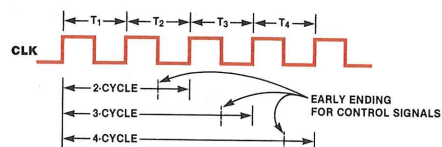


Figure 5. Variable Cycle Length

**Functional
Description**
(Continued)

Any of these interrupts cause an interrupt-pending status bit to be set, and each of them can optionally alter the DMA's interrupt vector. Due to the buffered constraint mentioned under "Modes of Operation," interrupts on Match at End of Block are caused by matches to the byte just prior to the last byte in the block.

The DMA shares the Z-80 Family's elaborate interrupt scheme, which provides fast interrupt service in real-time applications. In a Z-80 CPU environment, the DMA passes its internally modifiable 8-bit interrupt vector to the CPU, which adds an additional eight bits to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself.

**Pin
Description**

A₀-A₁₅. *System Address Bus* (output, 3-state). Addresses generated by the DMA are sent to both source and destination ports (main memory or I/O peripherals) on these lines.

BAI. *Bus Acknowledge In* (input, active Low). Signals that the system buses have been released for DMA control. In multiple-DMA configurations, the BAI pin of the highest priority DMA is normally connected to the Bus Acknowledge pin of the CPU. Lower-priority DMAs have their BAI connected to the BAO of a higher-priority DMA.

BAO. *Bus Acknowledge Out* (output, active Low). In a multiple-DMA configuration, this pin signals that no other higher-priority DMA has requested the system buses. BAI and BAO form a daisy chain for multiple-DMA priority resolution over bus control.

BUSRQ. *Bus Request* (bidirectional, active Low, open drain). As an output, it sends requests for control of the system address bus, data bus and control bus to the CPU. As an input when multiple DMAs are strung together in a priority daisy chain via BAI and BAO, it senses when another DMA has requested the buses and causes this DMA to refrain from bus requesting until the other DMA is finished. Because it is a bidirectional pin, there cannot be any buffers between this DMA and any other DMA. It can, however, have a buffer between it and the CPU because it is unidirectional into the CPU. A pull-up resistor is connected to this pin.

CE/WAIT. *Chip Enable and Wait* (input, active Low). Normally this functions only as a CE line, but it can also be programmed to serve a WAIT function. As a CE line from the

In this process, CPU control is transferred directly to the interrupt routine, so that the next instruction executed after an interrupt acknowledge is the first instruction of the interrupt routine itself.

Pulse Generation. External devices can keep track of how many bytes have been transferred by using the DMA's pulse output, which provides a signal at 256-byte intervals. The interval sequence may be offset at the beginning by 1 to 255 bytes.

The Interrupt line outputs the pulse signal in a manner that prevents misinterpretation by the CPU as an interrupt request, since it only appears when the Bus Request and Bus Acknowledge lines are both active.

CPU, it becomes active when $\overline{WR}$ and $\overline{IORQ}$ are active and the I/O port address on the system address bus is the DMA's address, thereby allowing a transfer of control or command bytes from the CPU to the DMA. As a WAIT line from memory or I/O devices, after the DMA has received a bus-request acknowledge from the CPU, it causes wait states to be inserted in the DMA's operation cycles thereby slowing the DMA to a speed that matches the memory or I/O device.

CLK. *System Clock* (input). Standard Z-80 single-phase clock at 2.5 MHz (Z-80 DMA) or 4.0 MHz (Z-80A DMA). For slower system clocks, a TTL gate with a large pullup resistor may be adequate to meet the timing and voltage level specification. For higher-speed systems, use a clock driver with an active pullup to meet the V_{IH} specification and risetime requirements.

D₀-D₇. *System Data Bus* (bidirectional, 3-state). Commands from the CPU, DMA status, and data from memory or I/O peripherals are transferred on these lines.

IEI. *Interrupt Enable In* (input, active High). This is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this DMA. Thus, this signal blocks lower-priority devices from interrupting while a higher-priority device is being serviced by its CPU interrupt service routine.

**Pin
Description**
(Continued)

INT/PULSE. *Interrupt Request* (output, active Low, open drain). This requests a CPU interrupt. The CPU acknowledges the interrupt by pulling its $\overline{\text{IORQ}}$ output Low during an M1 cycle. It is typically connected to the $\overline{\text{INT}}$ pin of the CPU with a pullup resistor and tied to all other $\overline{\text{INT}}$ pins in the system. This pin can also be used to generate periodic pulses to an external device. It can be used this way only when the DMA is bus master (i.e., the CPU's $\overline{\text{BUSRQ}}$ and $\overline{\text{BUSA}}\overline{\text{K}}$ lines are both Low and the CPU cannot see interrupts).

IORQ. *Input/Output Request* (bidirectional, active Low, 3-state). As an input, this indicates that the lower half of the address bus holds a valid I/O port address for transfer of control or status bytes from or to the CPU, respectively; this DMA is the addressed port if its $\overline{\text{CE}}$ pin and its $\overline{\text{WR}}$ or $\overline{\text{RD}}$ pins are simultaneously active. As an output, after the DMA has taken control of the system buses, it indicates that the lower half of the address bus holds a valid port address for another I/O device involved in a DMA transfer of data. When $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are both active simultaneously, an interrupt acknowledge is indicated.

M1. *Machine Cycle One* (input, active Low). Indicates that the current CPU machine cycle is an instruction fetch. It is used by the DMA to decode the return-from-interrupt instruction (RETI) (ED-4D) sent by the CPU. During two-byte instruction fetches, $\overline{\text{M1}}$ is active as each

opcode byte is fetched. An interrupt acknowledge is indicated when both $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are active.

MREQ. *Memory Request* (output, active Low, 3-state). This indicates that the address bus holds a valid address for a memory read or write operation. After the DMA has taken control of the system buses, it indicates a DMA transfer request from or to memory.

RD. *Read* (bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to read status bytes from the DMA's read registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled read from a memory or I/O port address.

RDY. *Ready* (input, programmable active Low or High). This is monitored by the DMA to determine when a peripheral device associated with a DMA port is ready for a read or write operation. Depending on the mode of DMA operation (Byte, Burst or Continuous), the RDY line indirectly controls DMA activity by causing the $\overline{\text{BUSRQ}}$ line to go Low or High.

WR. *Write* (bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to write control or command bytes to the DMA write registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled write to a memory or I/O port address.

**Internal
Structure**

The internal structure of the Z-80 DMA includes driver and receiver circuitry for interfacing with an 8-bit system data bus, a 16-bit system address bus, and system control lines (Figure 6). In a Z-80 CPU environment, the DMA can be tied directly to the analogous pins on the CPU (Figure 7) with no additional buffering, except for the $\overline{\text{CE}}/\overline{\text{WAIT}}$ line.

The DMA's internal data bus interfaces with the system data bus and services all internal logic and registers. Addresses generated from this logic for Ports A and B (source and destination) of the DMA's single transfer channel are multiplexed onto the system address bus.

Specialized logic circuits in the DMA are dedicated to the various functions of external bus interfacing, internal bus control, byte matching, byte counting, periodic pulse generation, CPU interrupts, bus requests, and address generation. A set of twenty-one writable control registers and seven readable status registers provides the means by which the CPU governs and monitors the activities of these logic circuits. All registers are eight bits wide, with double-byte information stored in adjacent registers. The two address counters (two bytes each) for Ports A and B are buffered by the two starting addresses.

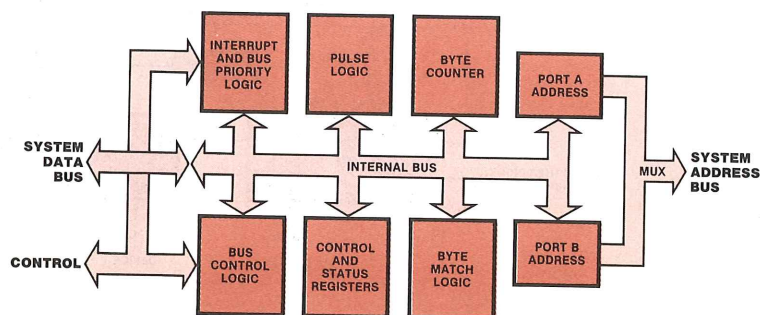


Figure 6. Block Diagram

Internal Structure (Continued)

The 21 writable control registers are organized into seven base-register groups, most of which have multiple registers. The base registers in each writable group contain both control/command bits and pointer bits that can be set to address other registers within the group. The seven readable status registers have no analogous second-level registers.

The registers are designated as follows, according to their base-register groups:

WR0-WR6 — Write Register groups 0 through 6 (7 base registers plus 14 associated registers)
RR0-RR6 — Read Registers 0 through 6

Writing to a register within a write-register group involves first writing to the base register, with the appropriate pointer bits set, then writing to one or more of the other registers within the group. All seven of the readable status registers are accessed sequentially according to a programmable mask contained in one of the writable registers. The section entitled "Programming" explains this in more detail.

A pipelining scheme is used for reading data in. The programmed block length is the number of bytes compared to the byte counter, which increments at the end of each cycle. In searches, data byte comparisons with the match byte are made during the read cycle of the next byte. Matches are, therefore, discovered only after the next byte is read in.

In multiple-DMA configurations, interrupt-request daisy chains are prioritized by the order in which their IEI and IEO lines are connected (Zilog Application Note 03-0041-01, *The Z-80 Family Program Interrupt Structure*). The

system bus, however, may not be pre-empted. Any DMA that gains access to the system buses keeps them until it is finished.

Write Registers

WR0	Base register byte Port A starting address (low byte) Port A starting address (high byte) Block length (low byte) Block length (high byte)
WR1	Base register byte Port A variable-timing byte
WR2	Base register byte Port B variable-timing byte
WR3	Base register byte Mask byte Match byte
WR4	Base register byte Port B starting address (low byte) Port B starting address (high byte) Interrupt control byte Pulse control byte Interrupt vector
WR5	Base register byte
WR6	Base register byte Read mask

Read Registers

RR0	Status byte
RR1	Byte counter (low byte)
RR2	Byte counter (high byte)
RR3	Port A address counter (low byte)
RR4	Port A address counter (high byte)
RR5	Port B address counter (low byte)
RR6	Port B address counter (high byte)

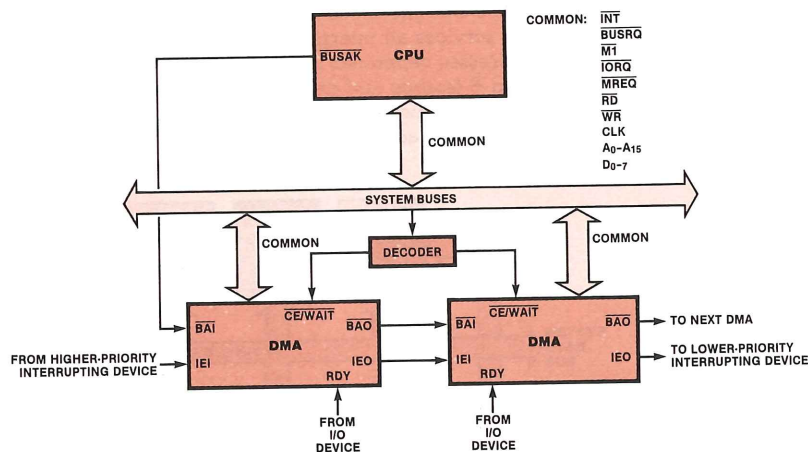


Figure 7. Multiple-DMA Interconnection to the Z-80 CPU

The Z-80 DMA has two programmable fundamental states: (1) an enabled state, in which it can gain control of the system buses and direct the transfer of data between ports, and (2) a disabled state, in which it can initiate neither bus requests nor data transfers. When the DMA is powered up or reset by any means, it is automatically placed into the disabled state. Program commands can be written to it by the CPU in either state, but this automatically puts the DMA in the disabled state, which is maintained until an enable command is issued by the CPU. The CPU must program the DMA in advance of any data search or transfer by addressing it as an I/O port and sending a sequence of control bytes using an Output instruction (such as OTIR for the Z-80 CPU).

Writing. Control or command bytes are written into one or more of the Write Register groups (WR0-WR6) by first writing to the base register byte in that group. All groups have base registers and most groups have additional associated registers. The associated registers in a group are sequentially accessed by first writing a byte to the base register containing register-group identification and pointer bits (1's) to one or more of that base register's associated registers.

This is illustrated in Figure 8. In this figure, the sequence in which associated registers within a group can be written to is shown by the vertical position of the associated registers. For example, if a byte written to the DMA contains the bits that identify WR0 (bits D0, D1 and D7), and also contains 1's in the bit positions that point to the associated "Port A Starting Address (low byte)" and "Port A Starting Address (high byte)," then the next two bytes written to the DMA will be stored in these two registers, in that order.

Reading. The Read Registers (RR0-RR6) are read by the CPU by addressing the DMA as an I/O port using an Input instruction (such as INIR for the Z-80 CPU). The readable bytes contain DMA status, byte-counter values, and port addresses since the last DMA reset. The

registers are always read in a fixed sequence beginning with RR0 and ending with RR6. However, the register read in this sequence is determined by programming the Read Mask in WR6. The sequence of reading is initialized by writing an Initiate Read Sequence or Set Read Status command to WR6. After a Reset DMA, the sequence must be initialized with the Initiate Read Sequence command or a Read Status command. The sequence of reading all registers that are not excluded by the Read Mask register must be completed before a new Initiate Read Sequence or Read Status command.

Fixed-Address Programming. A special circumstance arises when programming a destination port to have a fixed address. The load command in WR6 only loads a fixed address to a port selected as the source, not to a port selected as the destination. Therefore, a fixed destination address must be loaded by temporarily declaring it a fixed-source address and subsequently declaring the true source as such, thereby implicitly making the other a destination.

The following example illustrates the steps in this procedure, assuming that transfers are to occur from a variable-address source (Port A) to a fixed-address destination (Port B):

1. Temporarily declare Port B as source in WR0.
2. Load Port B address in WR6.
3. Declare Port A as source in WR0.
4. Load Port A address in WR6.
5. Enable DMA in WR6.

Figure 9 illustrates a program to transfer data from memory (Port A) to a peripheral device (Port B). In this example, the Port A memory starting address is 1050H and the Port B peripheral fixed address is 05H. Note that the data flow is 1001H bytes—one more than specified by the block length. The table of DMA commands may be stored in consecutive memory locations and transferred to the DMA with an output instruction such as the Z-80 CPU's OTIR instruction.

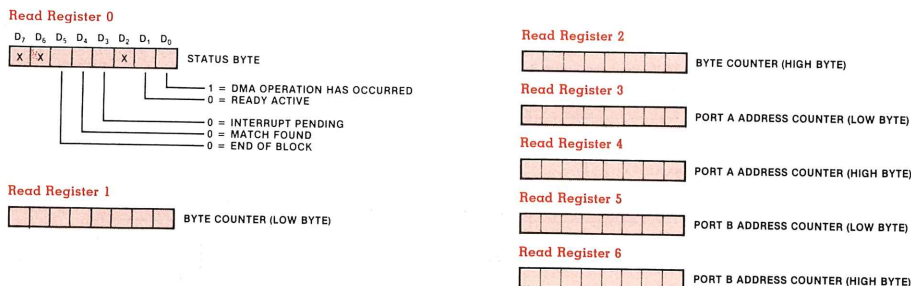


Figure 8a. Read Registers

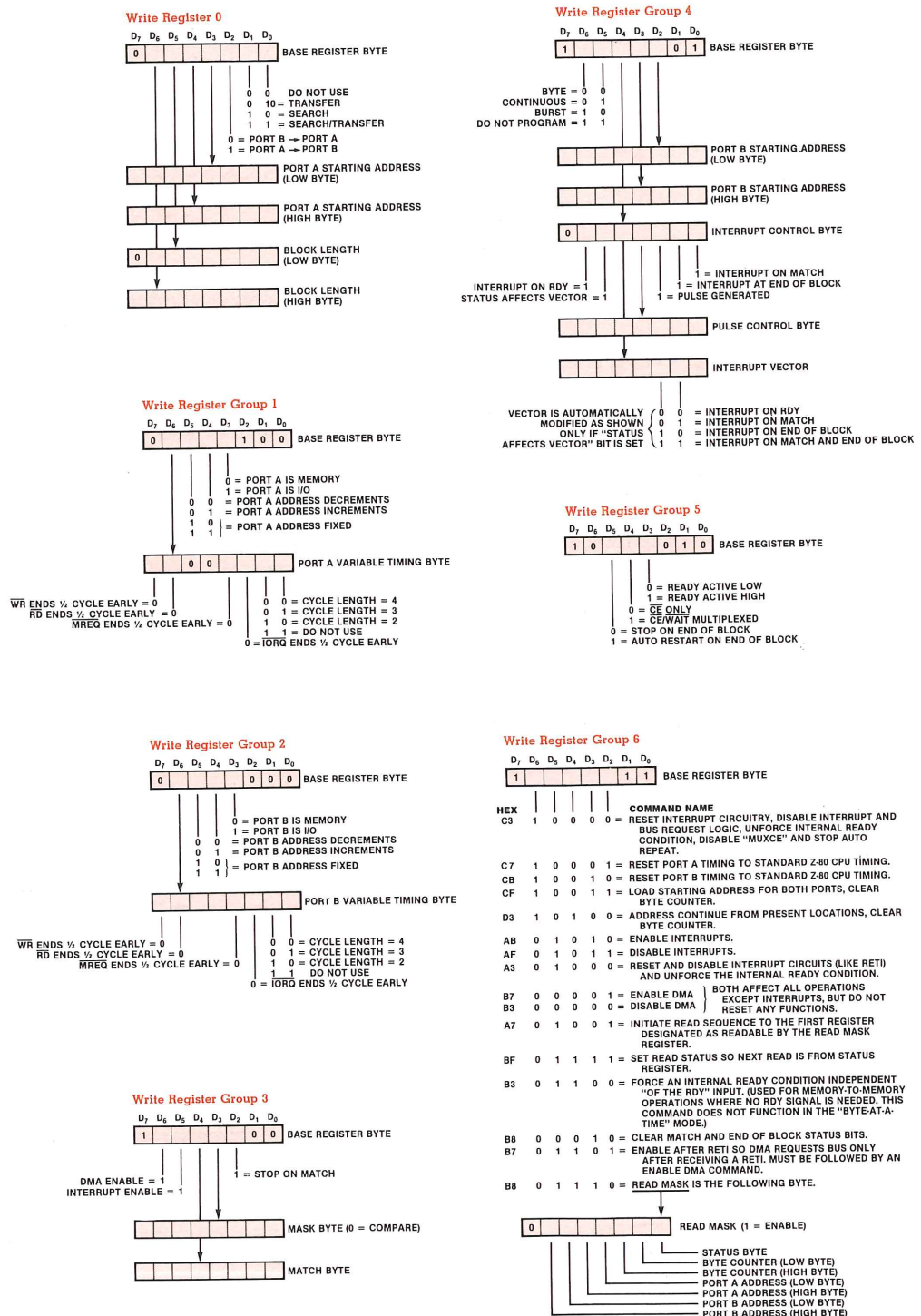


Figure 8b. Write Registers

Comments	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	HEX
WR0 sets DMA to receive block length. Port A starting address and temporarily sets Port B as source.	0	1 Block Length Upper Follows	1 Block Length Lower Follows	1 Port A Upper Address Follows	1 Port A Lower Address Follows	0 B → A Temporary for Loading B Address*	0 Transfer, No Search	1	79
Port A address (lower)	0	1	0	1	0	0	0	0	50
Port A address (upper)	0	0	0	1	0	0	0	0	10
Block length (lower)	0	0	0	0	0	0	0	0	00
Block length (upper)	0	0	0	1	0	0	0	0	10
WR1 defines Port A as memory with fixed incrementing address.	0	0 No Timing Follows	0 Address Changes	1 Address Increments	0 Port is Memory	1	0	0	14
WR2 defines Port B as peripheral with fixed address.	0	0 No Timing Follows	1 Fixed Address	0	1 Port is I/O	0	1	0	28
WR4 sets mode to Burst, sets DMA to expect Port B address.	1	1 Burst Mode	0	0 No Interrupt Control Byte Follows	0 No Upper Address	1 Port B Lower Address Follows	0	1	C5
Port B address (lower)	0	0	0	0	0	1	0	1	05
WR5 sets Ready active High.	1	0	0 No Auto Restart	0 No Wait States	1 RDY Active High	0	1	0	8A
WR6 loads Port B address and resets block counter.*	1	1	0	0	1	1	1	1	CF
WR0 sets Port A as source.*	0	0	0 No Address or Block Length Bytes	0	0	1 A → B	0 Transfer, No Search	1	05
WR6 loads Port A address and resets block counter.	1	1	0	0	1	1	1	1	CF
WR6 enables DMA to start operation.	1	0	0	0	0	1	1	1	87

NOTE: The actual number of bytes transferred is one more than specified by the block length.

*These entries are necessary only in the case of a fixed destination address.

Figure 9. Sample DMA Program

Inactive State Timing (DMA as CPU Peripheral)

In its disabled or inactive state, the DMA is addressed by the CPU as an I/O peripheral for write and read (control and status) operations. Write timing is illustrated in Figure 10.

Reading of the DMA's status byte, byte counter or port address counters is illustrated

in Figure 11. These operations require less than three T-cycles. The $\overline{CE}$, $\overline{IORQ}$ and $\overline{RD}$ lines are made active over two rising edges of CLK, and data appears on the bus approximately one T-cycle after they become active.

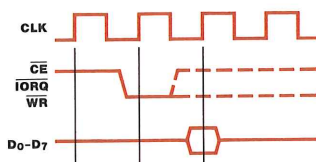


Figure 10. CPU-to-DMA Write Cycle

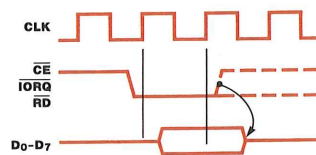


Figure 11. CPU-to-DMA Read Cycle

Active State Timing (DMA as Bus Controller)

Default Read and Write Cycles. By default, and after reset, the DMA's timing of read and write operations is exactly the same as the Z-80 CPU's timing of read and write cycles for memory and I/O peripherals, with one exception: during a read cycle, data is latched on the falling edge of T_3 and held on the data bus across the boundary between read and write cycles, through the end of the following write cycle.

Figure 12 illustrates the timing for memory-to-I/O port transfers and Figure 13 illustrates I/O-to-memory transfers. Memory-to-memory and I/O-to-I/O transfer timings are simply permutations of these diagrams.

The default timing uses three T-cycles for memory transactions and four T-cycles for I/O transactions, which include one automatically

inserted wait cycle between T_2 and T_3 . If the $\overline{CE}/\overline{WAIT}$ line is programmed to act as a $\overline{WAIT}$ line during the DMA's active state, it is sampled on the falling edge of T_2 for memory transactions and the falling edge of T_W for I/O transactions. If $\overline{CE}/\overline{WAIT}$ is Low during this time another T-cycle is added, during which the $\overline{CE}/\overline{WAIT}$ line will again be sampled. The duration of transactions can thus be indefinitely extended.

Variable Cycle and Edge Timing. The Z-80 DMA's default operation-cycle length for the source (read) port and destination (write) port can be independently programmed. This variable-cycle feature allows read or write cycles consisting of two, three or four T-cycles (more if Wait cycles are inserted), thereby increasing or decreasing the speed of all signals generated by the DMA. In addition,

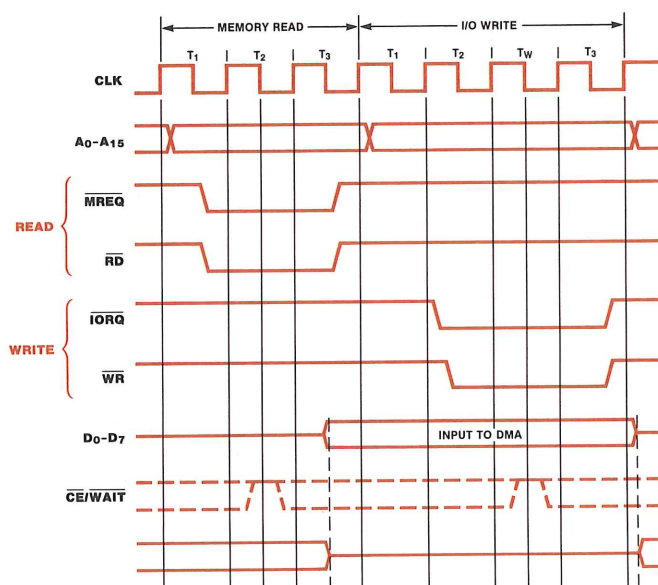


Figure 12. Memory-to-I/O Transfer

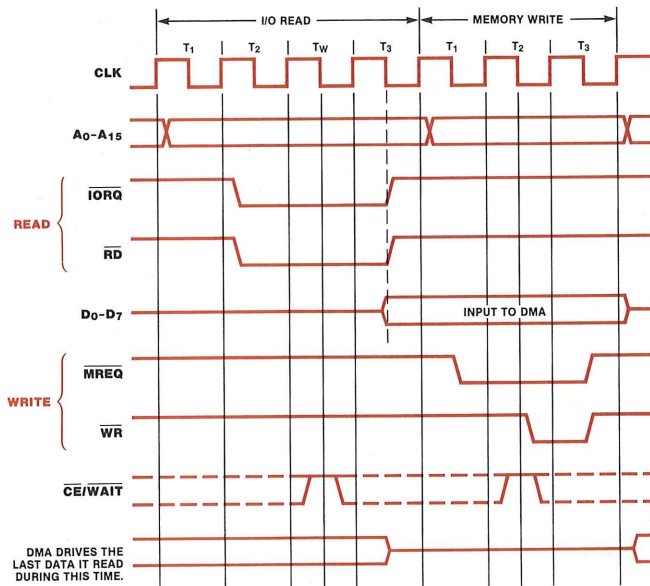


Figure 13. I/O-to-Memory Transfer

the trailing edges of the $\overline{\text{IORQ}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals can be independently terminated one-half cycle early. Figure 14 illustrates this.

In the variable-cycle mode, unlike default timing, $\overline{\text{IORQ}}$ comes active one-half cycle before $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$. $\overline{\text{CE/WAIT}}$ can be used to extend only the 3 or 4 T-cycle variable memory cycles and only the 4-cycle variable I/O cycle. The $\overline{\text{CE/WAIT}}$ line is sampled at the falling edge of T_2 for 3- or 4-cycle memory cycles, and at the falling edge of T_3 for 4-cycle I/O cycles.

During transfers, data is latched on the clock edge causing the rising edge of $\overline{\text{RD}}$ and held until the end of the write cycle.

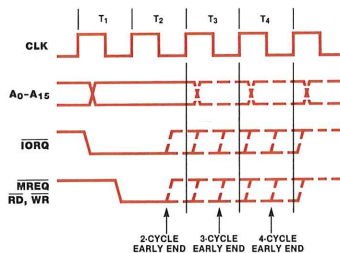


Figure 14. Variable-Cycle and Edge Timing

Figure 15 illustrates the bus request and acceptance timing. The RDY line, which may be programmed active High or Low, is sampled on every rising edge of CLK. If it is found to be active, and if the bus is not in use by any other device, the following rising edge of CLK drives $\overline{\text{BUSRQ}}$ low. After receiving $\overline{\text{BUSRQ}}$, the CPU acknowledges on the $\overline{\text{BAI}}$ input either directly or through a multiple-DMA daisy chain. When a Low is detected on $\overline{\text{BAI}}$ for two consecutive rising edges of CLK, the DMA will begin transferring data on the next rising edge of CLK.

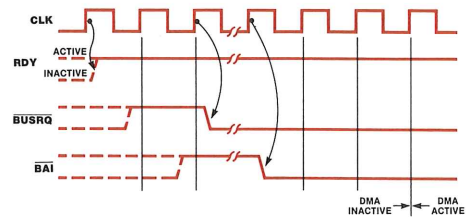


Figure 15. Bus Request and Acceptance

Bus Release Byte-at-a-Time. In Byte-at-a-Time mode, $\overline{\text{BUSRQ}}$ is brought High on the rising edge of CLK prior to the end of each read cycle (search-only) or write cycle (transfer and transfer/search) as illustrated in Figure 16. This is done regardless of the state of RDY. There is no possibility of confusion when a Z-80 CPU is used since the CPU cannot begin an operation until the following T-cycle. Most other CPUs are not bothered by this either, although note should be taken of it. The next bus request for the next byte will come after both $\overline{\text{BUSRQ}}$ and $\overline{\text{BAI}}$ have returned High.

Bus Release at End of Block. In Burst and Continuous modes, an end of block causes $\overline{\text{BUSRQ}}$ to go High usually on the same rising edge of CLK in which the DMA completes the transfer of the data block (Figure 17). The last byte in the block is transferred even if RDY goes inactive before completion of the last byte transfer.

Bus Release on Not Ready. In Burst mode, when RDY goes inactive it causes $\overline{\text{BUSRQ}}$ to go High on the next rising edge of CLK after the completion of its current byte operation (Figure 18). The action on $\overline{\text{BUSRQ}}$ is thus somewhat delayed from action on the RDY line. The DMA always completes its current byte operation in an orderly fashion before releasing the bus.

By contrast, $\overline{\text{BUSRQ}}$ is not released in Continuous mode when RDY goes inactive.

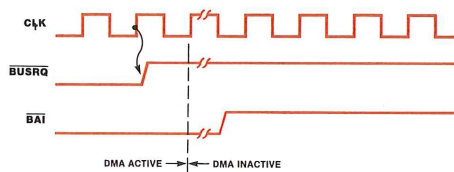


Figure 16. Bus Release (Byte-at-a-Time Mode)

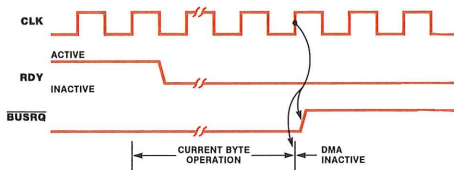


Figure 18. Bus Release When Not Ready (Burst Mode)

Instead, the DMA idles after completing the current byte operation, awaiting an active RDY again.

Bus Release on Match. If the DMA is programmed to stop on match in Burst or Continuous modes, a match causes $\overline{\text{BUSRQ}}$ to go inactive on the next DMA operation, i.e., at the end of the next read in a search or at the end of the following write in a transfer (Figure 19). Due to the pipelining scheme, matches are determined while the next DMA read or write is being performed.

The RDY line can go inactive after the matching operation begins without affecting this bus-release timing.

Interrupts. Timings for interrupt acknowledge and return from interrupt are the same as timings for these in other Z-80 peripherals. Refer to Zilog Application Note 03-0041-01 (*The Z-80 Family Program Interrupt Structure*).

Interrupt on RDY (interrupt before requesting bus) does not directly affect the $\overline{\text{BUSRQ}}$ line. Instead, the interrupt service routine must handle this by issuing the following commands to WR6:

1. Enable after Return From Interrupt (RETI) Command — Hex B7
2. Enable DMA — Hex 87
3. An RETI instruction that resets the Interrupt Under Service latch in the Z-80 DMA.

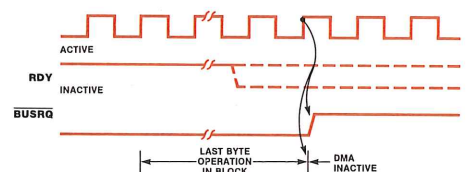


Figure 17. Bus Release at End of Block (Burst and Continuous Modes)

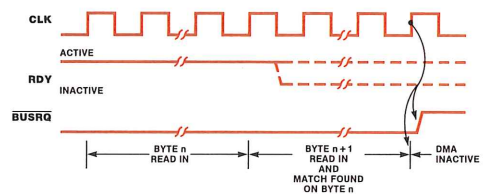


Figure 19. Bus Release on Match (Burst and Continuous Modes)

Absolute Maximum Ratings

Operating Ambient Temperature Under Bias . . . As Specified Under "Ordering Information"

Storage Temperature -65°C to $+150^{\circ}\text{C}$

Voltage On Any Pin with Respect to Ground -0.3 V to $+7\text{ V}$

Power Dissipation 1.5 W

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

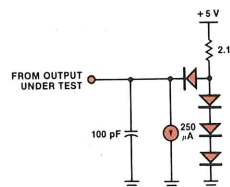
Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\text{ V} \leq V_{\text{CC}} \leq +5.25\text{ V}$
- $\text{GND} = 0\text{ V}$
- $0^{\circ}\text{C} \leq T_{\text{A}} \leq +70^{\circ}\text{C}$

All ac parameters assume a load capacitance of 100 pF max. Timing references between two

output signals assume a load difference of 50 pF max.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	$V_{\text{CC}} - 0.6$	5.5	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	5.5	V	
V_{OL}	Output Low Voltage		0.4	V	$I_{\text{OL}} = 3.2\text{ mA}$ for $\overline{\text{BUSRQ}}$ $I_{\text{OL}} = 2.0\text{ mA}$ for all others
V_{OH}	Output High Voltage	2.4		V	$I_{\text{OH}} = 250\text{ }\mu\text{A}$
I_{CC}	Power Supply Current				
	Z-80 DMA		150	mA	
	Z-80A DMA		200	mA	
I_{LI}	Input Leakage Current		10	μA	$V_{\text{IN}} = 0\text{ to }V_{\text{CC}}$
I_{LOH}	Tri-State Output Leakage Current in Float		10	μA	$V_{\text{OUT}} = 2.4\text{ to }V_{\text{CC}}$
I_{LOL}	Tri-State Output Leakage Current in Float		-10	μA	$V_{\text{OUT}} = 0.4\text{ V}$
I_{LD}	Data Bus Leakage Current in Input Mode		± 10	μA	$0 \leq V_{\text{IN}} \leq V_{\text{CC}}$

$V_{\text{CC}} = 5\text{ V} \pm 5\%$ unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		35	pF	Unmeasured Pins
C_{IN}	Input Capacitance		5	pF	Returned to Ground
C_{OUT}	Output Capacitance		10	pF	

$f = 1\text{ MHz}$, over specified temperature range.

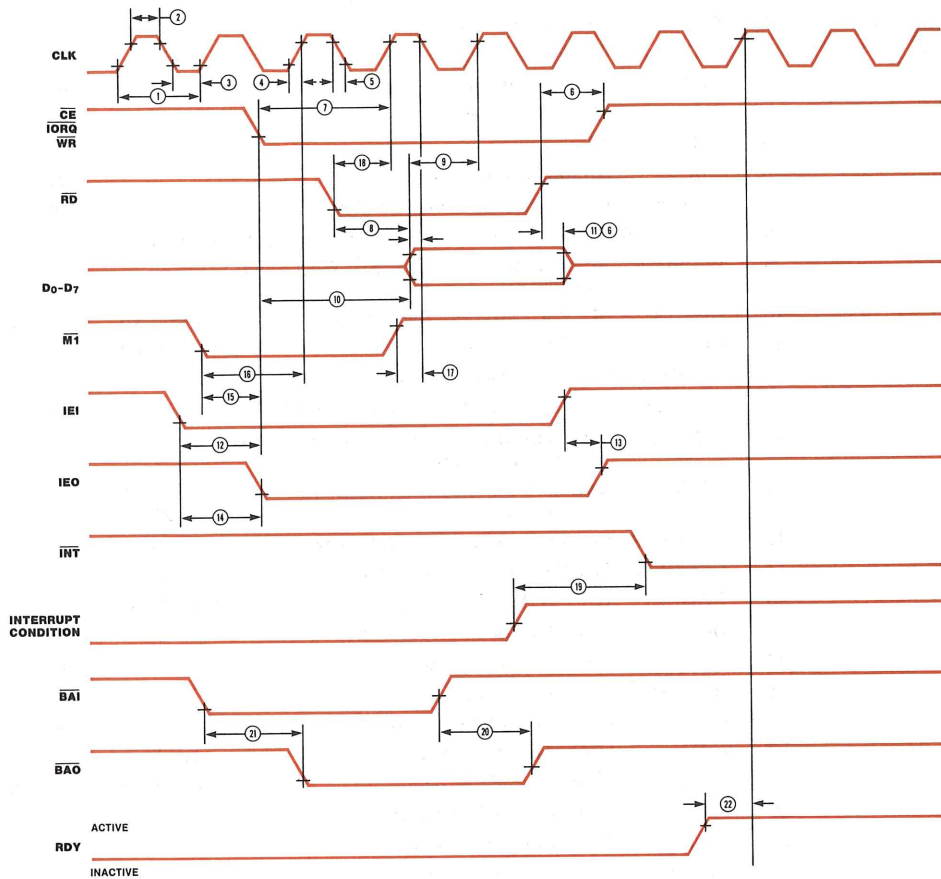
Inactive State AC Character- istics	Number	Symbol	Parameter	Z-80 DMA		Z-80A DMA		Unit
				Min	Max	Min	Max	
	1	T _c C	Clock Cycle Time	400	4000	250	4000	ns
	2	T _w Ch	Clock Width (High)	170	2000	105	2000	ns
	3	T _w C _l	Clock Width (Low)	170	2000	105	2000	ns
	4	T _r C	Clock Rise Time		30		30	ns
	5	T _f C	Clock Fall Time		30		30	ns
	6	T _h	Hold Time for Any Specified Setup Time	0		0		ns
	7	T _s C(Cr)	$\overline{\text{IORQ}}$, $\overline{\text{WR}}$, $\overline{\text{CE}}$ ↓ to Clock ↑ Setup	280		145		ns
	8	T _d DO(RDf)	$\overline{\text{RD}}$ ↓ to Data Output Delay		500		380	ns
	9	T _s WM(Cr)	Data In to Clock ↑ Setup ($\overline{\text{WR}}$ or $\overline{\text{M1}}$)	50		50		ns
	10	T _d Cf(DO)	$\overline{\text{IORQ}}$ ↓ to Data Out Delay (INTA Cycle)		340		160	ns
	11	T _d RD(Dz)	$\overline{\text{RD}}$ ↑ to Data Float Delay (output buffer disable)		160		110	ns
	12	T _s IEI(IORQ)	IEI ↓ to $\overline{\text{IORQ}}$ ↓ Setup (INTA Cycle)	140		140		ns
	13	T _d IEOr(IEIr)	IEI ↑ to IEO ↑ Delay		210		160	ns
	14	T _d IEOf(IEIf)	IEI ↓ to IEO ↓ Delay		190		130	ns
	15	T _d M1(IEO)	$\overline{\text{M1}}$ ↓ to IEO ↓ Delay (interrupt just prior to $\overline{\text{M1}}$ ↓)		300		190	ns
	16	T _s M1f(Cr)	$\overline{\text{M1}}$ ↓ to Clock ↑ Setup	210		90		ns
	17	T _s M1r(Cf)	$\overline{\text{M1}}$ ↑ to Clock ↓ Setup	20		-10		ns
	18	T _s RD(Cr)	$\overline{\text{RD}}$ ↓ to Clock ↑ Setup ($\overline{\text{M1}}$ Cycle)	240		115		ns
	19	T _d I(INT)	Interrupt Cause to $\overline{\text{INT}}$ ↓ Delay ($\overline{\text{INT}}$ generated only when DMA is inactive)		500		500	ns
	20	T _d BAIr(BAO _r)	$\overline{\text{BAI}}$ ↑ to $\overline{\text{BAO}}$ ↑ Delay		200		150	ns
	21	T _d BAIf(BAO _f)	$\overline{\text{BAI}}$ ↓ to $\overline{\text{BAO}}$ ↓ Delay		200		150	ns
	22	T _s RDY(Cr)	RDY Active to Clock ↑ Setup	150		100		ns

NOTE:

1. Negative minimum setup values mean that the first-mentioned event can come after the second-mentioned event.

**Inactive
State
AC
Character-
istics**
(Continued)

"1" "0"
CLOCK 4.2 V 0.8 V
OUTPUT 2.0 V 0.8 V
INPUT 2.0 V 0.8 V
FLOAT V = +0.5 V



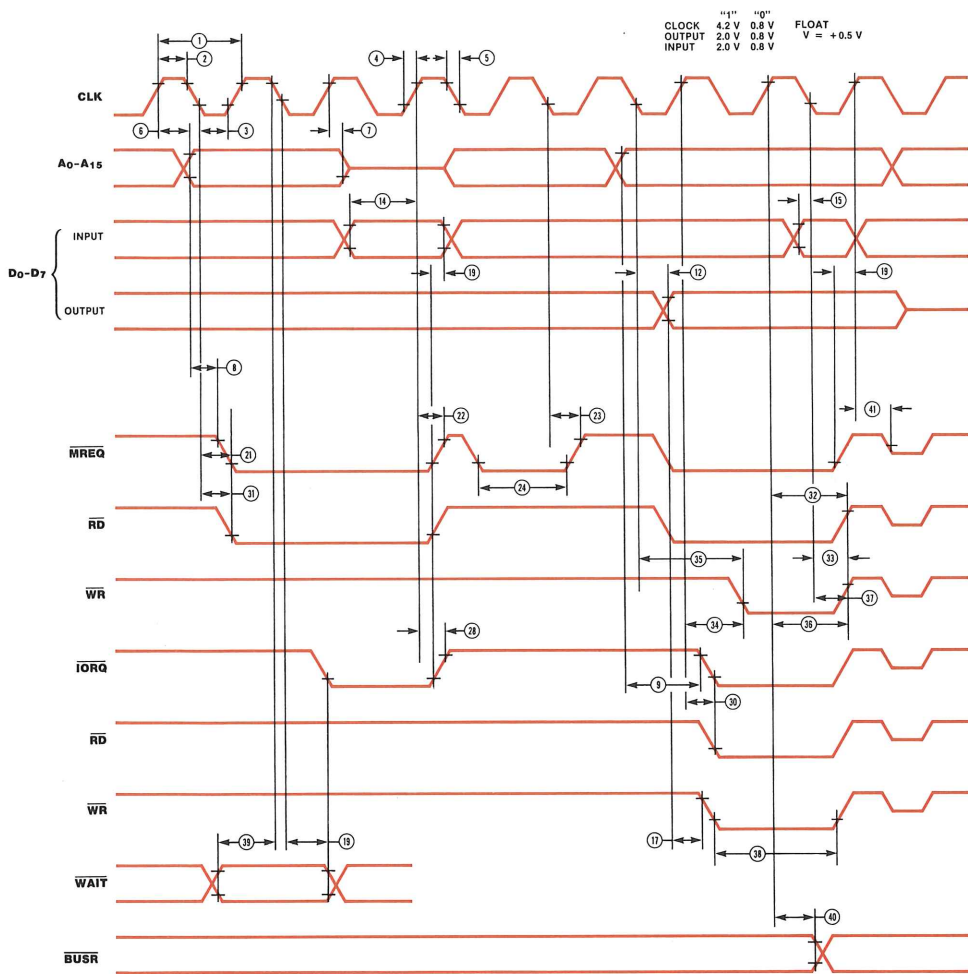
NOTE:
Signals in this diagram bear no relation to one another unless specifically noted as a numbered item.

Active State AC Character- istics	Number	Symbol	Parameter	Z-80 DMA		Z-80A DMA	
				Min(ns)	Max(ns)	Min(ns)	Max(ns)
	1	T _c C	Clock Cycle Time	400		250	
	2	T _w Ch	Clock Width (High)	180	2000	110	2000
	3	T _w C _l	Clock Width (Low)	180	2000	110	2000
	4	T _r C	Clock Rise Time		30		30
	5	T _f C	Clock Fall Time		30		30
	6	T _d A	Address Output Delay		145		110
	7	T _d C(A _z)	Clock ↑ to Address Float Delay		110		90
	8	T _s A(MREQ)	Address to $\overline{\text{MREQ}}$ ↓ Setup (Memory Cycle)	(2) + (5) - 75		(2) + (5) - 75	
	9	T _s A(IRW)	Address Stable to $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Setup (I/O Cycle)	(1) - 80		(1) - 70	
	*10	T _d RW(A)	$\overline{\text{RD}}$, $\overline{\text{WR}}$ ↑ to Addr. Stable Delay	(3) + (4) - 40		(3) + (4) - 50	
	*11	T _d RW(A _z)	$\overline{\text{RD}}$, $\overline{\text{WR}}$ ↑ to Addr. Float	(3) + (4) - 60		(3) + (4) - 45	
	12	T _d Cf(DO)	Clock ↓ to Data Out Delay		230		150
	*13	T _d Cr(D _z)	Clock ↑ to Data Float Delay (Write Cycle)		90		90
	14	T _s DI(Cr)	Data In to Clock ↑ Setup (Read cycle when rising edge ends read)	50		35	
	15	T _s DI(Cf)	Data In to Clock ↓ Setup (Read cycle when falling edge ends read)	60		50	
	*16	T _s DO(WfM)	Data Out to $\overline{\text{WR}}$ ↓ Setup (Memory Cycle)	(1) - 210		(1) - 170	
	17	T _s DO(WfI)	Data Out to $\overline{\text{WR}}$ ↓ Setup (I/O cycle)	100		100	
	*18	T _d Wr(DO)	$\overline{\text{WR}}$ ↑ to Data Out Delay	(3) + (4) - 80		(3) + (4) - 70	
	19	T _h	Hold Time for Any Specified Setup Time	0		0	
	20	T _d Cf(Mf)	Clock ↓ to $\overline{\text{MREQ}}$ ↓ Delay		100		85
	21	T _d Cr(Mr)	Clock ↑ to $\overline{\text{MREQ}}$ ↑ Delay		100		85
	22	T _d Cf(Mr)	Clock ↓ to $\overline{\text{MREQ}}$ ↑ Delay		100		85
	23	T _w M _l	$\overline{\text{MREQ}}$ Low Pulse Width	(1) - 40		(1) - 30	
	*24	T _w M _h	$\overline{\text{MREQ}}$ High Pulse Width	(2) + (5) - 30		(2) + (5) - 20	
	25	T _d Cr(I _f)	Clock ↑ to $\overline{\text{IORQ}}$ ↓ Delay		90		75
	26	T _d Cr(I _r)	Clock ↑ to $\overline{\text{IORQ}}$ ↑ Delay		100		85
	*27	T _d Cf(I _r)	Clock ↓ to $\overline{\text{IORQ}}$ ↑ Delay		110		85
	28	T _d Cr(R _f)	Clock ↑ to $\overline{\text{RD}}$ ↓ Delay		100		85
	29	T _d Cf(R _f)	Clock ↓ to $\overline{\text{RD}}$ ↓ Delay		130		95
	30	T _d Cr(R _r)	Clock ↑ to $\overline{\text{RD}}$ ↑ Delay		100		85
	31	T _d Cf(R _r)	Clock ↓ to $\overline{\text{RD}}$ ↑ Delay		110		85
	32	T _d Cr(W _f)	Clock ↑ to $\overline{\text{WR}}$ ↓ Delay		80		65
	33	T _d Cf(W _f)	Clock ↓ to $\overline{\text{WR}}$ ↓ Delay		90		80
	34	T _d Cr(W _r)	Clock ↑ to $\overline{\text{WR}}$ ↑ Delay		100		80
	35	T _d Cf(W _r)	Clock ↓ to $\overline{\text{WR}}$ ↑ Delay		100		80
	36	T _w W _l	$\overline{\text{WR}}$ Low Pulse Width	(1) - 40		(1) - 30	
	37	T _s WA(Cf)	$\overline{\text{WAIT}}$ to Clock ↓ Setup	70		70	
	38	T _d Cr(B)	Clock ↑ to $\overline{\text{BUSRQ}}$ Delay		100		100
	39	T _d Cr(I _z)	Clock ↑ to $\overline{\text{IORQ}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$ Float Delay		100		80

NOTES:

1. Numbers in parentheses are other parameter-numbers in this table; their values should be substituted in equations.
2. All equations imply DMA default (standard) timing.
3. Data must be enabled onto data bus when $\overline{\text{RD}}$ is active.
4. Asterisk (*) before parameter number means the parameter is not illustrated in the AC Timing Diagrams.

**Active
State
AC
Character-
istics**
(Continued)



NOTE:
Signals in this diagram bear no relation to one another unless specifically noted as a numbered item.

Z-UPC/U Universal Peripheral Controller



NEW
1980

Product Brief

1

Preliminary

February 1980

Features

- Complete slave microcomputer, for distributed-processing use.
- Unmatched power of Z8 architecture, instruction set.
- Three programmable I/O ports, two with 2-wire handshake, or any combination of data and control lines.
- Six levels of priority interrupts to Z-UPC.
- Two programmable 8-bit counter/timers with 6-bit prescalers.
- 256 byte register file, accessible by both master CPU and Z-UPC, as allocated by Z-UPC program.
- 2K bytes of on-chip program ROM for efficiency, versatility.

Description

The Z-UPC Universal Peripheral Controller is a distributed microcomputer that performs the three basic interfacing functions needed to interface a CPU with peripherals: device control by ROM-resident internal software, data manipulation, such as reformatting or arithmetic, and data buffering in internal registers.

The Z-UPC is similar to the Z8 microcomputer and uses the Z8 instruction set. Under

program control, its three 8-line I/O ports can be tailored to the needs of its user. Permanently configured as a single-chip controller with 2K bytes of internal ROM, the Z-UPC executes instructions in 2.2 μ s average using a 4-MHz clock source. Its register file contains 256 bytes, of which 234 are general-purpose registers, 19 are status and control registers, and three are port registers.

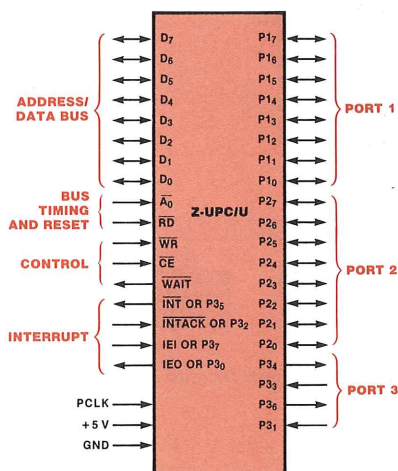


Figure 1. Pin Functions

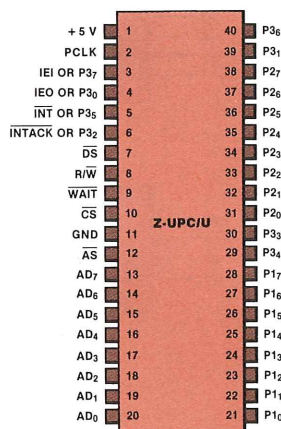


Figure 2. Pin Assignments

Description
(Continued)

The Z-UPC Universal Peripheral Controller is an intelligent device that generates all the control signals peripheral devices need. Because it does off-line arithmetic, translates data before transmitting, and buffers data, the Z-UPC unburdens the master CPU, thereby increasing the overall speed and efficiency of the system in which it resides.

Based upon the Z8 microcomputer architecture, the Z-UPC offers fast execution time, efficient use of memory, and sophisticated interrupt, I/O, and bit manipulation. Its powerful and extensive instruction types, combined with its efficient internal register addressing scheme, not only speeds program execution, but also efficiently packs program into the on-chip ROM.

A unique characteristic of the Z-UPC is its register file, which contains I/O port and control registers that can be accessed both by the Z-UPC program and by its associated master CPU. This results in byte efficiency, programming efficiency, and address space efficiency because Z-UPC instructions can operate directly on I/O data without moving it to and from an accumulator. It also allows the Z-UPC user to allocate as data buffer between the CPU and

the peripheral all register space not in use as accumulators, address pointers, index registers, or stack. Registers not used as buffer are protected against CPU access. The register file is divided into 16 groups of 16 working registers each. A register pointer uses fast, short-format instructions to access any one of these groups quickly, resulting in fast and easy task switching. Two-way communication between the master CPU and the register file is facilitated by another pointer that positions 16 interface registers anywhere within the register file. These registers are accessed directly by both the master CPU and the slave Z-UPC. Four more registers, similarly accessed, convey control and status information.

All of Zilog's daisy-chained priority interrupt system can be implemented in the Z-UPC under software control, or the Z-UPC can be programmed to function in a polled environment. In all, the Z-UPC has 24 pins that can be dedicated to I/O functions. Grouped logically into three 8-line ports, they can be programmed in many combinations of inputs, outputs, and bidirectional lines, with or without handshake and with push-pull or open-drain outputs.

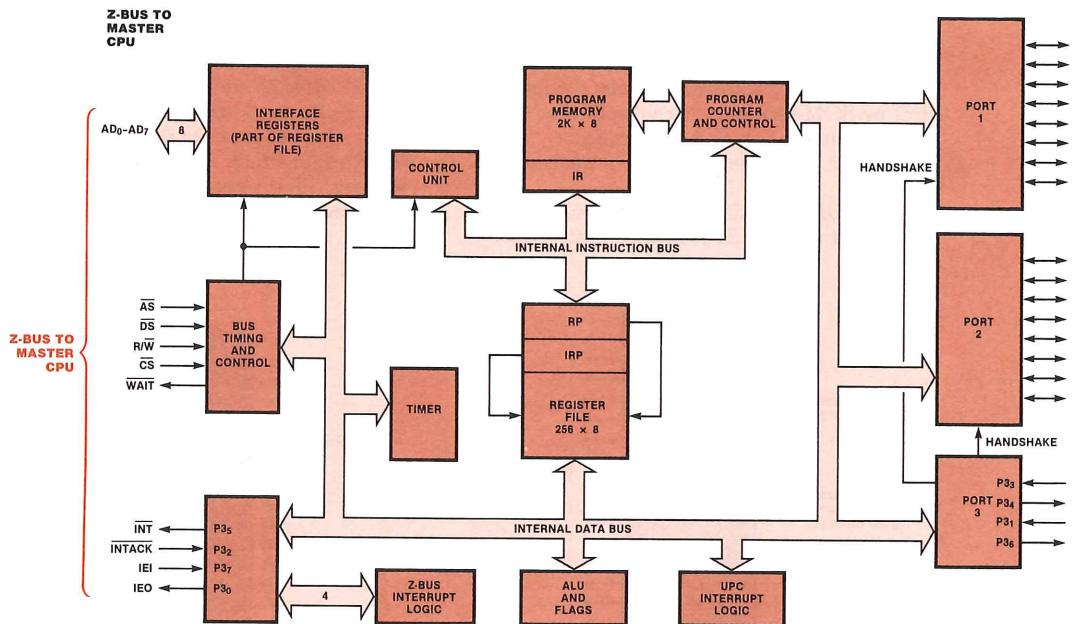


Figure 3. Functional Block Diagram

Z-CIO/U Counter/Timer and Parallel I/O Unit



NEW
1980

Product Brief

1

Preliminary

February 1980

Features

- Two independent 8-bit double-buffered bidirectional I/O ports plus a special-purpose 4-bit I/O port.
- Four handshake modes including IEEE-488.
- Wait/Request line for high speed data transfer.
- Three independent 16-bit counters.
- All registers read/write and directly addressable.
- Flexible pattern recognition logic, programmable as 16-input interrupt controller.

Description

The Z-CIO Counter/Timer and Parallel I/O element is a general purpose peripheral circuit that satisfies most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications. The use of the device is simplified by making all internal registers (command, status, and data) readable and (except for status bits) writable. Also, each register is given its own unique address so it can be accessed directly—no special sequential operations are required.

Either 8-bit I/O port can be a handshake

byte port or a bit port. In the bit mode, data direction is programmable bit by bit. In the handshake mode, the ports can be input, output, or bidirectional, and they may be linked to form a 16-bit port. The four handshake modes include IEEE-488, interlocked (for interfacing to a Z-UPC, Z-FIO or another Z-CIO), strobed and pulsed. The pulsed mode connects one counter/timer with the handshake logic for interfacing a mechanical device such as a printer. The 4-bit port provides handshake controls, special controls (Wait/Request) or general-purpose I/O.

The counter/timer section contains three 16-bit counters, two of which can be software-configured as a 32-bit counter/timer. Up to

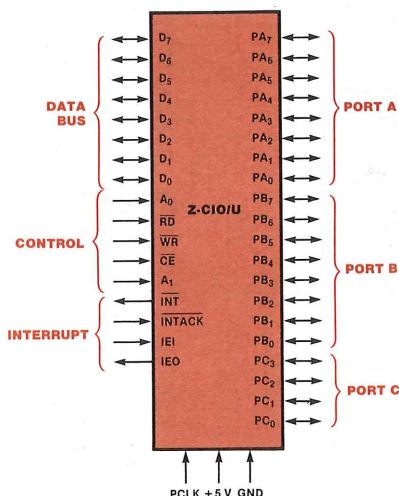


Figure 1. Pin Functions

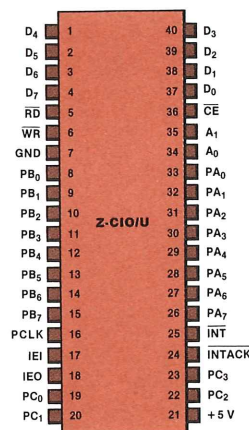


Figure 2. Pin Assignments

Description (Continued)

four I/O lines for each counter are available for direct external control and status information. All counters have a programmable output duty cycle, continuous or single-cycle operation, and the counting process can be programmed to be either retrigged or nonretrigged.

Figure 3 shows how the Z-CIO is used. The two general purpose 8-bit ports are similar. They can be programmed as handshake driven, double-buffered ports (input, output, or bidirectional) or as control ports in which the direction of each bit is individually programmable. Port B can also be specified to provide external access for two of the counter/timers. Each port includes pattern recognition logic allowing interrupt generation when a specified pattern is detected. The pattern recognition logic can be programmed so that the port functions like a priority interrupt controller.

To control these capabilities, each port contains 13 registers. Three of these, the input, output, and buffer registers, are data path registers. Two others, the mode specification and handshake specification registers, define the mode of the port and specify what handshake to use, if any. The reference pattern for the pattern recognition logic is defined in three registers, the pattern polarity, pattern transition, and pattern mask registers. The detailed characteristics of each bit path (for example, the direction of data flow, or whether a path is inverting or noninverting) are programmed using the data path polarity, data direction, and special I/O control registers. The primary control and status bits are grouped in a single register so that after the ports are configured initially, only this register

need be accessed often. One register contains the interrupt vector associated with each port. To facilitate initialization, the port logic is designed so that if a capability of the port is not required the registers associated with that capability are ignored and need not be programmed.

The function of port C depends upon the roles of ports A and B. Port C provides handshake lines for the other two when required. Any bits of port C not so used can be used as I/O lines or as external access to the third counter/timer.

Besides the data input and output registers, three registers are needed. These specify the details of each bit path: data path polarity, data direction, and special I/O control.

The three counter/timers are all identical. Each is composed of a 16-bit down-counter, a 16-bit time constant register (which holds the value loaded into the down-counter), a 16-bit current count register (used to read the contents of the down-counter), and two 8-bit registers for control and status (the mode select and control registers). All three share a common vector register.

Each counter/timer can be programmed as either counter or timer. Up to four port I/O lines can be designated as external access lines for it. The lines are: Counter Input, Gate Input, Trigger Input, and Counter/Timer Output. Three different counter/timer output duty cycles are available: pulse, one-shot, or square wave. The operation of the counter/timer can be specified to be either single cycle or continuous. The counting sequence may be retrigged or nonretrigged, under program control.

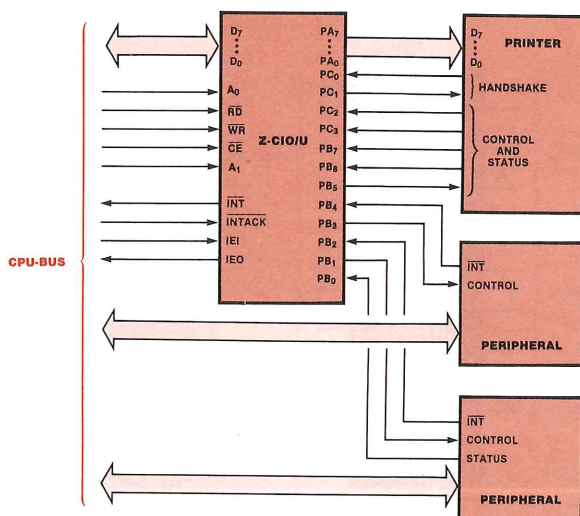


Figure 3. Functional Block Diagram

Z-FIO FIFO Input/Output Interface Unit



NEW
1980

Product Brief

1

Preliminary

February 1980

Features

- Asynchronous bidirectional FIFO buffer, used with most major microprocessors as CPU/CPU or CPU/peripheral interface.
- Interlocked or IEEE-488 handshake port mode; Empty, Full, and Wait/Request lines for high-speed data transfer.
- 128 x 8 organization, expandable to any width; cascadable to any depth.
- Preset byte count in FIO buffer can interrupt CPU.
- All registers directly addressable.
- Vectored/non-vectored interrupts on pattern/status match, over/underflow error, buffer status.

Description

The Z-FIO is a general-purpose microprocessor interface that provides elastic buffering between asynchronous CPUs in a parallel-processor network or between CPU and peripheral circuits. The Z-FIO can interface a Z-bus microprocessor or any other major processor to another microprocessor or to a peripheral circuit or port.

In Z8000 systems, the FIO furthers distributed-processor operation because it can interconnect components or subsystems operating at different speeds. Also, it can increase system throughput by transferring

words as well as bytes. This bidirectional device accepts data and holds it until it can be used by another device in the system. In most I/O transactions, introducing a 128-deep buffer cuts interrupt servicing overhead by two orders of magnitude.

The Z-FIO greatly facilitates system throughput by moving variable-size blocks under either direct memory access or interrupt control—an especially important consideration when fast peripheral circuits need interfacing. Complete status information is also provided for operation in polled environments.

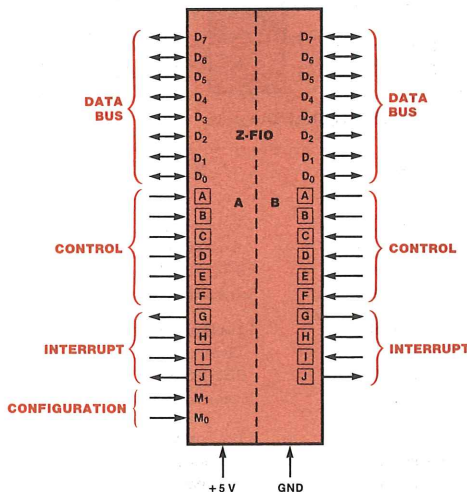


Figure 1. Pin Functions

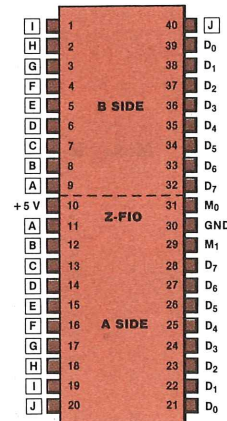


Figure 2. Pin Assignments

Pin Assignments	Z-Bus Low Byte	Z-Bus High Byte	General Purpose	Interlocked HS Port*	IEEE 488 HS Port*
A	REQ/WT	REQ/WT	REQ/WT	RFD/DAV	RFD/DAV
B	DMASTB	DMASTB	DACK	ACKIN	DAV/DAC
C	DS	DS	RD	FULL	DAC/RFD
D	R/W	R/W	WR	EMPTY	EMPTY
E	CS	CS	CS	CLEAR	CLEAR
F	AS	AS	C/D	DATA DIR	DATA DIR
G	INTACK	A ₀	INTACK	IN ₀	IN ₀
H	IEO	A ₁	IEO	OUT ₁	OUT ₁
I	IEI	A ₂	IEI	IN ₂	IN ₂
J	INT	A ₃	INT	OUT ₃	OUT ₃

* B side only. See table below.

Description (Continued)

The internal functions of the Z-FIO are shown in the block diagram (Figure 3). It is made up of two sides that are identical except for programming. The side programmed by pins M₀ and M₁ is called the A side; the side programmed by bits SL₀ and SL₁ is called the B side. Common to both, and situated between the two sides, is the 128 × 8 RAM used for data storage, two 7-bit counters, and various registers. The RAM is capable of simultaneous, independent read and write operations. This means, for example, that the A side CPU can write a byte of data into the FIO without disturbing a simultaneous read operation by the B side CPU. The outputs of the read and write counters are used to address the buffer RAM,

and also are fed into a subtractor to determine the current number of bytes in the memory. This number can be read by either CPU from a status register dedicated to each side. Another programmable register is compared against the status register to generate an interrupt and/or start and stop DMA transfers. A pair of port registers allows for communication between CPUs, bypassing the main buffer memory.

Operating Modes. The Z-FIO has twelve different programmable modes. (Table below.) The states of two package pins determine the mode of operation of the A side, and the B side is programmed by two bits (SL₀ and SL₁) in one of the A side control registers.

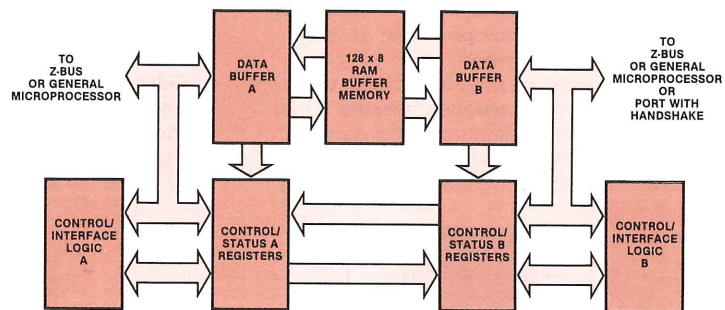


Figure 3. Functional Block Diagram

Operating Modes	Mode	M ₁	M ₀	SL ₁	SL ₀	A	B
	0	0	0	0	0	Z-Bus Low Byte	Z-Bus Low Byte
	1	0	0	0	1	Z-Bus Low Byte	General μP
	2	0	0	1	0	Z-Bus Low Byte	IEEE 488 HS
	3	0	0	1	1	Z-Bus Low Byte	Interlocked HS
	4	0	1	0	0	Z-Bus High Byte	Z-Bus High Byte
	5	0	1	0	1	Z-Bus High Byte	General μP
	6	0	1	1	0	Z-Bus High Byte	IEEE 488 HS
	7	0	1	1	1	Z-Bus High Byte	Interlocked HS
	8	1	0	0	0	General μP	Z-Bus Low Byte
	9	1	0	0	1	General μP	General μP
	10	1	0	1	0	General μP	IEEE 488 HS
	11	1	0	1	1	General μP	Interlocked HS

2

Z8000 Family

Zilog

Zilog Z8000 Family



The Art of Staying a Generation Ahead

February 1980

If you know much about microprocessors, you know that Zilog became a technology leader by introducing the 8-bit Z-80 Family. The soundness of that family design has been amply proven, and its popularity is still growing.

Naturally, you expect Zilog to come up with more than just an extension of an 8-bit architecture to 16 bits. And you are right. The new Z8000 Family of microprocessor components is a whole generation ahead. Now you can design advanced concepts from the mainframe and minicomputer worlds into microcomputer systems that do more at less cost than ever before.

You won't find anything like this anywhere but Zilog. The advanced architecture, high throughput, intelligent peripherals and system flexibility of the Z8000 Family are sure to make it the most popular microprocessor of the 1980s. Want to find out why? Get acquainted with the family. Read on.

System Flexibility. The Z-Family spans the gulf between simple stand-alone computers and complex multiple-processor systems. Your system can grow as your application matures or expands.

Even the smallest Z8000 systems offer high throughput and easy programming far superior to any existing microprocessor alternative. In mid-range applications, Z8000 components offer much more powerful solutions to the design problems of word processing, intelligent terminals, data communications, instrumentation and process control. And in a complex network of multiple processors, smart peripheral components, small local memories and a large common memory, the Z8000 Family provides performance and versatility exceeding that of much larger—and far more expensive—minicomputers.

Higher Throughput. Reduced Cost. The powerful instruction set, high execution speed, regular architecture and numerous special features of the Z8000 CPU dramatically increase system throughput. Intelligent Z8000 peripheral controllers unburden the CPU and boost throughput even more.

Simply put, the Z8000 Family offers more for less money. The Z8000 CPU gives mid-range minicomputer performance at microprocessor cost. At component prices, Z8000 peripheral controllers perform complex system functions that previously required an entire PC board. Memory costs are reduced by Z-Bus memories that require no external logic, and by the compact code and moderate clock rate of the Z8000 CPU.

The Z8000 Family is designed for multiple-processor operation—an economical way of greatly increasing system performance. Many special features for multiple Z8000 CPUs facilitate the design of multiple-processor systems that share access to a common memory. The Memory Management Unit can dynamically relocate code and protect memory areas. The Z-UPC Universal Peripheral Controller, a complete slave microcomputer, can manipulate data off-line. Asynchronous parts of multiple-processor systems can be joined by the Z-FIO FIFO Interface Unit. Z-Bus compatible memories make small local memory for dedicated CPUs an affordable item.

2

An Unmatched CPU. The Z8000 CPU is far more than a wider data path, more registers, more data types, more addressing modes, more instructions and more addressing space. It brings the big-machine concepts to the level of microprocessors. The instruction set of the Z8000 CPU is more powerful than that of many minicomputers and incorporates features previously found only in large mainframes. Its general-register architecture avoids bottlenecks associated with dedicated or implied registers. Special features support parallel processors, operating systems and compilers. For example, its ability to operate in both the System and Normal modes separates user programs from the operating system for better software security and modularity.

The Z8000 CPU is also a very fast machine. Its throughput is greater than that of any other 16-bit microprocessor and many minicomputers. And the Z8000 CPU achieves this at a moderate 4 MHz clock rate that allows you the choice of slow-access, low-cost memories.

How to Manage Your Memory Better. Modern trends are toward systems that have large and growing memories, multiple users, complex programs and requirements for effective system security. These design problems pose questions not sufficiently answered by other microprocessor families.

Another example of the Z-Family commitment to advanced architectural concepts, the MMU provides flexibility in code segment relocation and sophistication in memory protection found nowhere else in the microprocessor world. This unique device encourages modular software development—a necessary trend as programs reach new levels of complexity.

You are free from specifying where information is actually located in the physical memory because the MMU makes software addresses totally independent from the actual physical memory address. Some existing microprocessor CPUs do have internal CPU relocation registers, but they are dedicated and support few segments, and these CPUs restrict memory protection. Not true for the MMU. Various MMU configurations can randomly relocate all 128 segments output by the Z8000 CPU in all its six addressing spaces and with various translation tables for each space.

But the MMU is far more than a relocation device. It offers you a host of memory protection features that allow the system to protect its software from unwanted uses and users. Segments can be specified as read-only to protect them from being overwritten, as system-only to protect the operating system from inadvertent user access, as execute-only, and so on. A write warning zone is especially useful in stack operations so the operating system can deal with growing stacks.

Peripheral Problem Solvers. Z8000 peripheral components are not dumb I/O circuits. They perform intelligent, complicated tasks on their own. They unburden the CPU, reduce bus traffic and increase system throughput. Complex system tasks that previously required burdensome conglomerations of MSI, can now be handled off-line by Z-Bus peripherals with little CPU overhead.

Multifunction Z-Bus peripherals are extensively programmable, so each can be precisely tailored to its application. All share common interrupt and bus-request structures, as well as an I/O command structure that addresses up to 64 internal registers. Z-Bus peripherals can be operated in priority-interrupt or polled environments. They offer multiple channels to minimize chip count.

Counting, timing and parallel I/O problems seem less tiresome with the **Z8036 Counter and Parallel I/O device**. It has three 16-bit counter/timers, three I/O ports and can even double as a programmable interrupt-priority controller. Data communications are neatly handled by the **Z8030 Serial Communication Controller**, a dual-channel multi-protocol component that supports all popular communications formats. Direct memory access is amply supported by the **Z8016 DMA Controller**, a fast dual-channel device that enhances the addressing power of the Z8000 CPU in stand-alone or parallel-processor environments. You can interface a variety of CRT displays with the **Z8052 CRT Controller**, which offers a vertical or horizontal split screen, oversize alphanumerics, smooth scrolling, and numerous other features. General-purpose control and data-manipulation problems are smoothly solved by the **Z8034 Universal Peripheral Controller**, a complete off-line microcomputer-on-a-chip with three I/O ports. This processor executes the same friendly, capable instruction set as our Z8 Microcomputer. Bits and pieces of asynchronous parallel-processing systems are interconnected by the **Z8038 FIO Interface Unit**, a surprisingly flexible device that can interface any major microprocessor and most peripherals to the Z-Bus. Its buffer depth can be expanded without limit using the **Z8060 FIFO**. Small local memory is affordable because the **Z6132 4K x 8 Quasi-Static Z-Bus RAM** requires no external interface circuitry, and it refreshes itself.

Z8001 CPU Z8002 CPU



**NEW
1980**

Product Specification

Preliminary

February 1980

Features

- Regular, easy-to-use architecture
- Instruction set more powerful than many minicomputers
- Directly addresses 8M bytes
- Eight user-selectable addressing modes
- Seven data types that range from bits to 32-bit long words and word strings
- System and Normal operating modes
- Separate code, data and stack spaces
- Sophisticated interrupt structure
- Resource-sharing capabilities for multi-processing systems
- Multi-programming support
- Compiler support
- Memory management and protection provided by Z8010 Memory Management Unit
- 32-bit operations, including signed multiply and divide
- Z-Bus compatible
- 4 MHz clock rate

2

General Description

The Z8000 is an advanced high-end 16-bit microprocessor that spans a wide variety of applications ranging from simple stand-alone computers to complex parallel-processing systems. Essentially, a monolithic minicomputer central processing unit, the Z8000 CPU is characterized by an instruction set more powerful than many minicomputers; resources abundant in registers, data types, addressing modes and addressing range; and a regular architecture that enhances throughput by avoiding critical bottlenecks such as implied or dedicated registers.

CPU resources include sixteen 16-bit general-purpose registers, seven data types that range from bits to 32-bit long words and word strings, and eight user-selectable addressing modes. The 110 distinct instruction types can be combined with the various data types and addressing modes to form a powerful set of 414 instructions. Moreover, the instruction set exhibits a high degree of regularity: most instructions can use any of the five main addressing modes and can operate on byte, word and long-word data types.

The CPU can operate in either the system or normal modes. The distinction between these two modes permits privileged operations, thereby improving operating system organization and implementation. Multiprogramming is supported by the "atomic" Test and Set instruction; multiprocessing by a combination

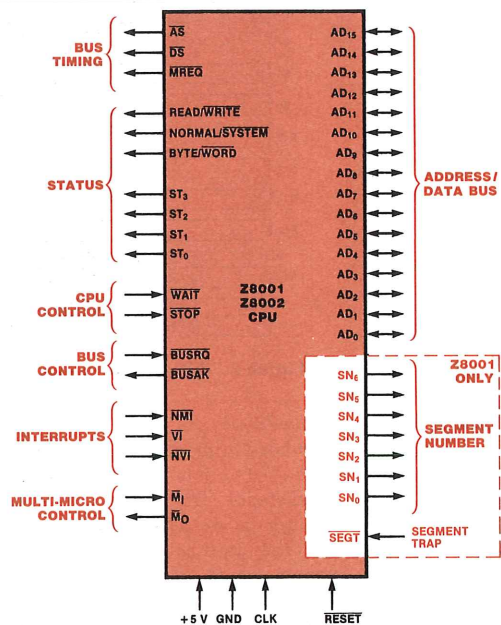


Figure 1. Z8000 CPU Pin Functions

of instruction and hardware features; and compilers by multiple stacks, special instructions and addressing modes.

**General
Description**
(Continued)

The Z8000 CPU is offered in two versions: the Z8001 48-pin segmented CPU and the Z8002 40-pin non-segmented CPU. The main difference between the two is in addressing range. The Z8001 can directly address 8 megabytes of memory; the Z8002 directly addresses 64 kilobytes. The two operating modes—system and normal—and the distinction between code, data and stack spaces within each mode allows memory extension up to 48 megabytes for the Z8001 and 384 kilobytes for the Z8002.

To meet the requirements of complex, memory-intensive applications, a companion

memory-management device is offered for the Z8001. The Z8010 Memory Management Unit manages the large address space by providing features such as segment relocation and memory protection. The Z8001 can be used with or without the Z8010. If used by itself, the Z8001 still provides an 8 megabyte direct addressing range, extendable to 48 megabytes.

The Z8001, Z8002 and Z8010 are fabricated with high-density, high-performance scaled n-channel silicon-gate depletion-load technology, and are housed in dual in-line packages.

**Register
Organization**

The Z8000 CPU is a register-oriented machine that offers sixteen 16-bit general-purpose registers and a set of special system registers. All general-purpose registers can be used as accumulators and all but one as index registers or memory pointers.

Register flexibility is created by grouping and overlapping multiple registers (Figures 2

and 3). For byte operations, the first eight 16-bit registers (R0...R7) are treated as sixteen 8-bit registers (RL0, RH0, ..., RL7, RH7). The sixteen 16-bit registers are grouped in pairs (RR0...RR14) to form 32-bit long-word registers. Similarly, the register set is grouped in quadruples (RQ0...RQ12) to form 64-bit registers.

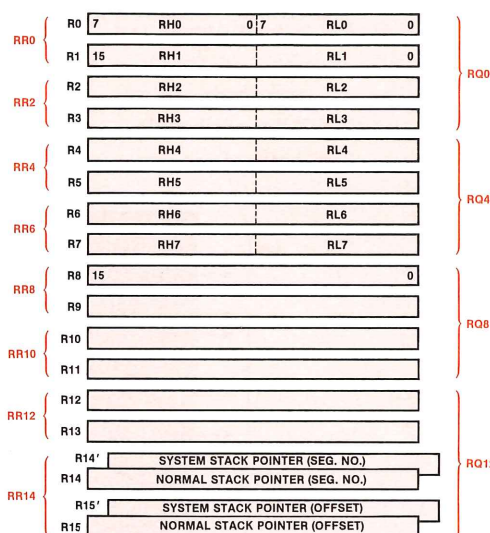


Figure 2. Z8001 General-Purpose Registers

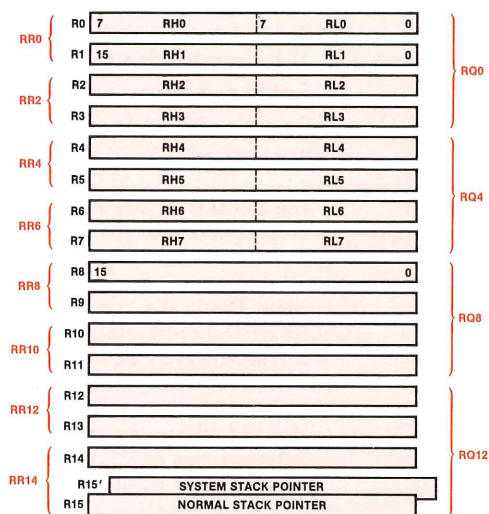


Figure 3. Z8002 General-Purpose Registers

Stacks

The Z8001 and Z8002 can use stacks located anywhere in memory. Call and Return instructions as well as interrupts and traps use implied stacks. The distinction between normal and system stacks separates system information from the application program information. Two stack pointers are available: the system stack pointer and the normal stack pointer. Because they are part of the general-purpose register

group, the user can manipulate the stack pointers with any instruction available for register operations.

In the Z8001, register pair RR14 is the implied stack pointer. Register R14 contains the 7-bit segment number and R15 contains the 16-bit offset. In the Z8002, register R15 is the implied 16-bit stack pointer.

Refresh

The Z8000 CPU contains a counter that can be used to automatically refresh dynamic memory. The refresh counter register consists of a 9-bit row counter, a 6-bit rate counter and an enable bit (Figure 4). The 9-bit row counter can address up to 256 rows and is incremented by two each time the rate counter reaches end-of-count. The rate counter determines the time between successive refreshes. It consists of a programmable 6-bit modulo-n prescaler

(n = 1 to 64), driven at one-fourth the CPU clock rate. The refresh period can be programmed from 1 to 64 μ s with a 4 MHz clock. Refresh can be disabled by programming the refresh enable/disable bit.



Figure 4. Refresh Counter

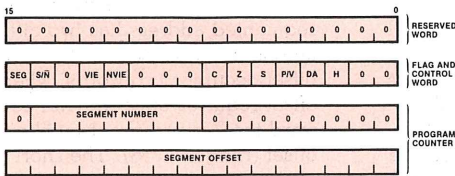
Program Status Information

This group of status registers contains the program counter, flags and control bits. When an interrupt or trap occurs, the entire group is saved and a new program status group is loaded.

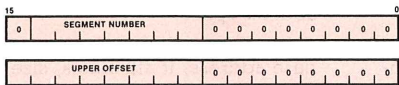
Figure 5 illustrates how the program status groups of the Z8001 and Z8002 differ. In the non-segmented Z8002, the program status group consists of two words: the program counter (PC), and the flag and control word (FCW). In the segmented Z8001, the program

status group consists of four words: a two-word program counter, the flag and control word and an unused word reserved for future use. Seven bits of the first PC word designate one of the 128 memory segments. The second word supplies the 16-bit offset that designates a memory location within the segment.

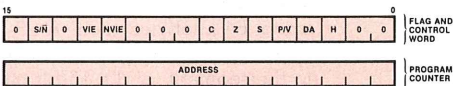
With the exception of the segment enable bit in the Z8001 program status group, the flags and control bits are the same for both CPUs.



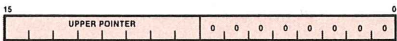
Z8001 Program Status Registers



Z8001 Program Status Area Pointer



Z8002 Program Status Registers



Z8002 Program Status Area Pointer

Figure 5. Z8000 CPU Special Registers

Interrupt and Trap Structure

The Z8000 provides a very flexible and powerful interrupt and trap structure. Interrupts are external asynchronous events requiring CPU attention, and are generally triggered by peripherals needing service. Traps are synchronous events resulting from the execution of certain instructions. Both are processed in a similar manner by the CPU.

The CPU supports three types of interrupts (non-maskable, vectored and non-vectored) and four traps (system call, unimplemented instruction, privileged instructions and segmentation trap). The vectored and non-vectored interrupts are maskable. Of the four traps, the only external one is the segmentation trap, which is generated by the Z8010.

The remaining traps occur when instructions limited to the system mode are used in the normal mode, or as a result of the System Call instruction, or for an unimplemented instruction. The descending order of priority for traps

and interrupts is: internal traps, non-maskable interrupt, segmentation trap, vectored interrupt and non-vectored interrupt.

When an interrupt or trap occurs, the current program status is automatically pushed on the system stack. The program status consists of the processor status (PC and FCW) plus a 16-bit identifier. The identifier contains the reason or source of the trap or interrupt. For internal traps, the identifier is the first word of the trapped instruction. For external traps or interrupts, the identifier is the vector on the data bus read by the CPU during the interrupt-acknowledge or trap-acknowledge cycle.

After saving the current program status, the new program status is automatically loaded from the program status area in system memory. This area is designated by the program status area pointer (PSAP).

Data Types Z8000 instructions can operate on bits, BCD digits (4 bits), bytes (8 bits), words (16 bits), long words (32 bits), byte strings and word strings (up to 64 kilobytes long). Bits can be set, reset and tested; digits are used in BCD arithmetic operations; bytes are used for characters or small integer values; words are used for integer values, instructions and non-segmented addresses; long words are used for

Segmentation and Memory Management High-level languages, sophisticated operating systems, large programs and data bases, and decreasing memory prices are all accelerating the trend toward larger memory requirements in microcomputer systems. The Z8001 meets this requirement with an eight

Segmented Addressing A segmented addressing space—compared with linear addressing—is closer to the way a programmer uses memory because each procedure and data space resides in its own segment. The 8 megabytes of Z8001 addressing space is divided into 128 relocatable segments up to 64 kilobytes each. A 23-bit segmented address uses a 7-bit segment address to point to the segment, and a 16-bit offset to address any location relative to the beginning of the segment. The two parts of the segmented address may be manipulated separately. The segmented Z8001 can run any code written for the non-segmented Z8002 in any one of its 128 segments, provided it is set to the non-segmented mode.

Memory Management The addresses manipulated by the programmer, used by instructions and output by the Z8001 are called *logical* addresses. The Memory Management Unit takes the logical addresses and transforms them into the *physical* addresses required for accessing the memory (Figure 6). This address transformation process is called relocation. Segment relocation makes user software addresses independent of the physical memory so the user is freed from specifying where information is actually located in the physical memory.

The relocation process is transparent to user software. A translation table in the Memory Management Unit associates the 7-bit segment number with the base address of the physical memory segment. The 16-bit offset is added to the physical base address to obtain the actual physical address. The system may dynamically reload translation tables as tasks are created, suspended or changed.

In addition to supporting dynamic segment relocation, the Memory Management Unit also provides segment protection and other segment management features. The protection features prevent illegal uses of segments, such as writing into a write-protected zone.

Each Memory Management Unit stores 64 segment entries that consist of the segment

long integer values and segmented addresses. All data elements except strings can reside either in registers or memory. Strings are stored in memory only.

The basic data element is the byte. The number of bytes used when manipulating a data element is either implied by the operation or—for strings and multiple register operations—explicitly specified in the instruction.

megabyte addressing space. This large address space is directly accessed by the CPU using a segmented addressing scheme and can be managed by the Z8010 Memory Management Unit.

In hardware, segmented addresses are contained in a register pair or long-word memory location. The segment number and offset can be manipulated separately or together by all the available word and long-word operations.

When contained in an instruction, a segmented address has two different representations: long offset and short offset. The long offset occupies two words, whereas the short offset requires only one and combines in one word the 7-bit segment number with an 8-bit offset (range 0–256). The short offset mode allows very dense encoding of addresses and minimizes the need for long addresses required by direct accessing of this large address space.

base address, it attributes, size and status. Segments are variable in size from 256 bytes to 64 kilobytes in increments of 256 bytes. Pairs of Management Units support the 128 segment numbers available for each of the six CPU address spaces. Within an address space, several Management Units can be used to create multiple translation tables.

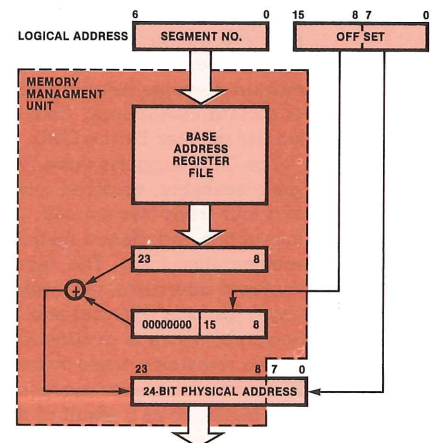


Figure 6. Logical-to-Physical Address Transformation

Addressing Modes

The information included in Z8000 instructions consists of the function to be performed, the type and size of data elements to be manipulated and the location of the data elements. Locations are designated by register addresses, memory addresses or I/O addresses. The addressing mode of a given instruction defines the address space it references and the method used to compute the address itself. Addressing modes are explicitly specified or implied by the instruction.

Figure 4 illustrates the eight addressing modes: Register (R), Immediate (IM), Indirect Register (IR), Direct Address (DA), Indexed (X), Relative Address (RA), Base Address (BA) and Base Indexed (BX). In general, an addressing mode explicitly specifies either register address space or memory address space. Program memory address space and I/O address space are usually implied by the instruction.

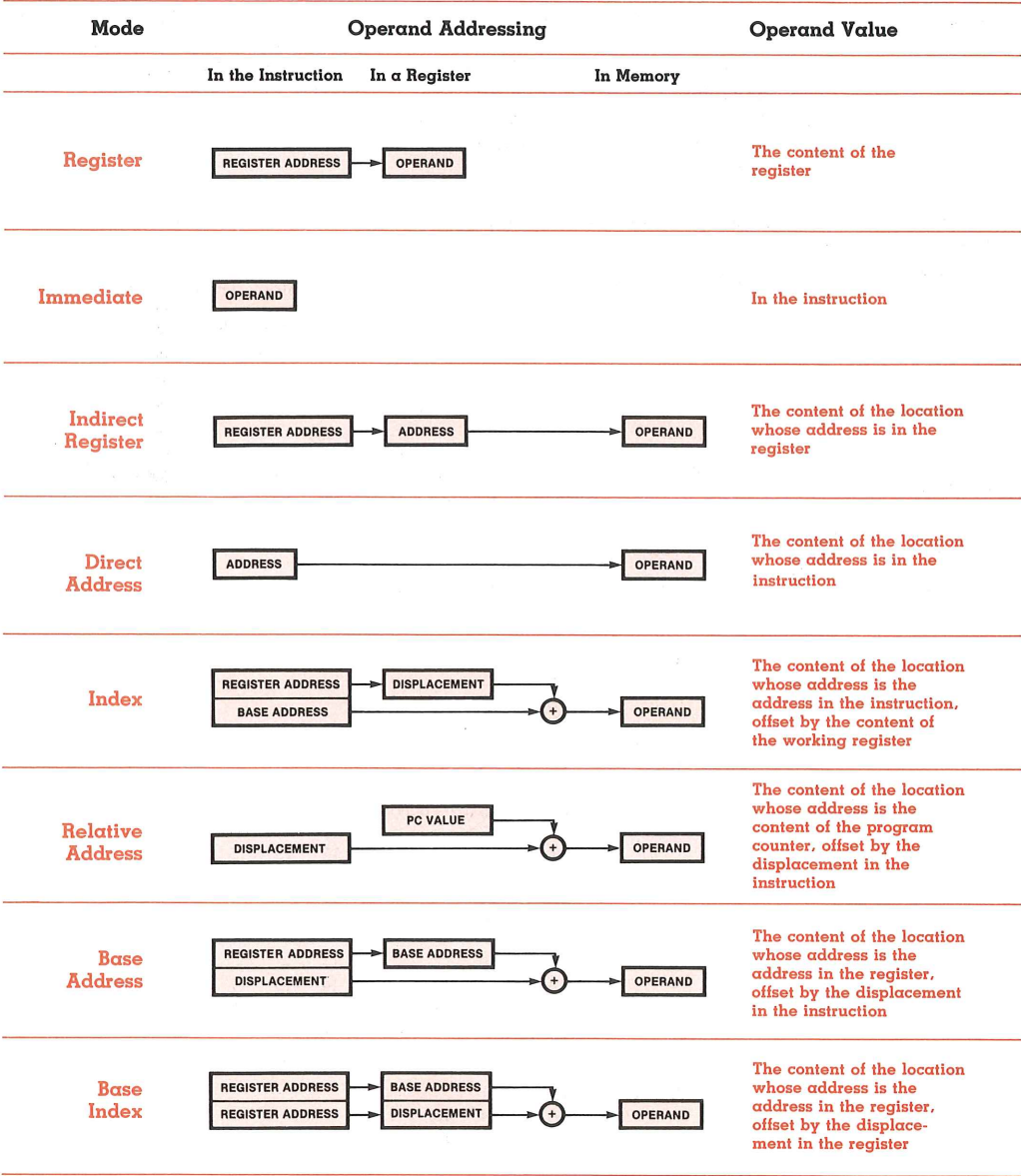


Figure 7. Addressing Modes

Input/Output	A set of I/O instructions performs 8-bit or 16-bit transfers between the CPU and I/O devices. I/O devices are addressed with a 16-bit I/O port address. The I/O port address is similar to a memory address; however, I/O address space is not part of the memory address space. I/O port and memory addresses coexist on the same bus lines and they are distinguished by the status outputs.	Two types of I/O instructions are available: standard and special. Each has its own address space. Standard I/O instructions include a comprehensive set of In, Out and Block I/O instructions for both bytes and words. Special I/O instructions are used for loading and unloading the Memory Management Unit. The status information distinguishes between standard and special I/O references.
Multi-Micro-Processor Support	Multi-microprocessor systems are supported in hardware and software. A pair of CPU pins is used in conjunction with certain instructions to coordinate multiple microprocessors. The Multi-Micro Out pin issues a request for the resource, while the Multi-Micro In pin is used to recognize the state of the resource. Thus, any CPU in a multiple microprocessor system can exclude all other asynchronous CPUs from a critical shared resource.	Multi-microprocessor systems are supported in software by the instructions Multi-Micro Request, Test Multi-Micro In, Set Multi-Micro Out and Reset Multi-Micro Out. In addition, the eight megabyte CPU address space is beneficial in multiple microprocessor systems that have large memory requirements.
Instruction Set Summary	The Z8000 provides the following types of instructions: ■ Load and Exchange ■ Arithmetic ■ Logical ■ Program Control	■ Bit Manipulation ■ Rotate and Shift ■ Block Transfer and String Manipulation ■ Input/Output ■ CPU Control

Load and Exchange	Mnemonics	Operands	Addr. Modes	Clock Cycles*						Operation
				Word, Byte			Long Word			
				NS	SS	SL	NS	SS	SL	
CLR CLRB	dst	R	7	-	-				Clear dst ← 0	
		IR	8	-	-					
		DA	11	12	14					
		X	11	12	15					
EX EXB	R, src	R	6	-	-				Exchange R ↔ src	
		IR	12	-	-					
		DA	15	16	18					
		X	16	16	19					
LD LDB LDL	R, src	R	3	-	-	5	-	-	Load into Register R ← src	
		IM	7	-	-	11	-	-		
		IM	5 (byte only)							
		IR	7	-	-	11	-	-		
		DA	9	10	12	12	13	15		
		X	10	10	13	13	13	16		
		BA	14	-	-	17	-	-		
		BX	14	-	-	17	-	-		
LD LDB LDL	dst, R	IR	8	-	-	11	-	-	Load into Memory (Store) dst ← R	
		DA	11	12	14	14	15	17		
		X	12	12	15	15	15	18		
		BA	14	-	-	17	-	-		
		BX	14	-	-	17	-	-		
LD LDB	dst, IM	IR	11	-	-				Load Immediate into Memory dst ← IM	
		DA	14	15	17					
		X	15	15	18					

* NS = Non-Segmented SS = Segmented Short Offset SL = Segmented Long Offset

**Load and
Exchange**
(Continued)

Load and Exchange (Continued)	Mnemonics	Operands	Addr. Modes	Clock Cycles						Operation	
				Word, Byte			Long Word				
				NS	SS	SL	NS	SS	SL		
LDA	R, src	DA	12	13	15					Load Address R ← source address	
		X	13	13	16						
		BA	15	-	-						
		BX	15	-	-						
LDAR	R, src	RA	15	-	-					Load Address Relative R ← source address	
LDK	R, src	IM	5	-	-					Load Constant R ← n (n = 0 ... 15)	
LDM	R, src, n	IR	11	-	-					Load Multiple R ← src (n consecutive words) (n = 1 ... 16)	
		DA	14	15	17	} + 3 n					
		X	15	15	18						
LDM	dst, R, n	IR	11	-	-					Load Multiple (Store Multiple) dst ← R (n consecutive words) (n = 1 ... 16)	
		DA	14	15	17	} + 3 n					
		X	15	15	18						
LDR LDRB LDRL	R, src	RA	14	-	-	17	-	-	Load Relative R ← src (range -32768 ... +32767)		
LDR LDRB LDRL	dst, R	RA	14	-	-	17	-	-	Load Relative (Store Relative) dst ← R (range -32768 ... +32767)		
POP POPL	dst, IR	R	8	-	-	12	-	-	Pop dst ← IR Autoincrement contents of R		
		IR	12	-	-	19	-	-			
		DA	15	16	18	22	23	25			
		X	16	16	19	23	23	26			
PUSH PUSHL	IR, src	R	9	-	-	12	-	-	Push Autodecrement contents of R IR ← src		
		IM	12	-	-	-	-	-			
		IR	13	-	-	20	-	-			
		DA	13	14	16	20	21	23			
		X	14	14	17	21	21	24			
Arithmetic	ADC ADCB	R, src	R	5	-	-					Add with Carry R ← R + src + carry
	ADD ADDB ADDL	R, src	R	4	-	-	8	-	-	Add R ← R + src	
			IM	7	-	-	14	-	-		
			IR	7	-	-	14	-	-		
			DA	9	10	12	15	16	18		
			X	10	10	13	16	16	19		
	CP CPB CPL	R, src	R	4	-	-	8	-	-	Compare with Register R - src	
			IM	7	-	-	14	-	-		
			IR	7	-	-	14	-	-		
			DA	9	10	12	15	16	18		
			X	10	10	13	16	16	19		
	CP CPB	dst, IM	IR	11	-	-					Compare with Immediate dst - IM
			DA	14	15	17					
			X	15	15	18					
	DAB	dst	R	5	-	-					Decimal Adjust
	DEC DECB	dst, n	R	4	-	-					Decrement by n dst ← dst - n (n = 1 ... 16)
IR			11	-	-						
DA			13	14	16						
X			14	14	17						

Arithmetic
(Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles						Operation
			Word, Byte			Long Word			
			NS	SS	SL	NS	SS	SL	
DIV DIVL	R, src	R	95	-	-	723	-	-	Divide
		IM	95	-	-	723	-	-	Word: $R_{n+1} \leftarrow R_{n,n+1} \div \text{src}$
		IR	95	-	-	723	-	-	$R_n \leftarrow \text{remainder}$
		DA	96	97	99	724	725	727	Long Word: $R_{n+2,n+3} \leftarrow R_{n\dots,n+3} \div \text{src}$
		X	97	97	100	725	725	728	$R_{n,n+1} \leftarrow \text{remainder}$
EXTS EXTSB EXTSL	dst	R	11	-	-	11	-	-	Extend Sign
									Extend sign of low order half of dst through high order half of dst
INC INCB	dst, n	R	4	-	-				Increment by n
		IR	11	-	-				dst $\leftarrow$ dst + n
		DA	13	14	16				(n = 1 ... 16)
		X	14	14	17				
MULT MULTL	R, src	R	70	-	-	282*	-	-	Multiply
		IM	70	-	-	282*	-	-	Word: $R_{n,n+1} \leftarrow R_{n+1} \cdot \text{src}$
		IR	70	-	-	282*	-	-	Long Word: $R_{n\dots,n+3} \leftarrow R_{n+2,n+3} \cdot \text{src}$
		DA	71	72	74	283*	284*	286*	* Plus seven cycles for each 1 in the
		X	72	72	75	284*	284*	287*	multiplicand
NEG NEGB	dst	R	7	-	-				Negate
		IR	12	-	-				dst $\leftarrow$ 0 - dst
		DA	15	16	18				
		X	16	16	19				
SBC SBCB	R, src	R	5	-	-				Subtract with Carry
									R $\leftarrow$ R - src - carry
SUB SUBB SUBL	R, src	R	4	-	-	8	-	-	Subtract
		IM	7	-	-	14	-	-	R $\leftarrow$ R - src
		IR	7	-	-	14	-	-	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	

Logical

AND ANDB	R, src	R IM IR DA X	4 7 7 9 10	- - - 10 10	- - - 12 13	AND R ← R AND src			
COM COMB	dst	R IR DA X	7 12 15 16	- - 16 16	- - 18 19	Complement dst ← NOT dst			
OR ORB	R, src	R IM IR DA X	4 7 7 9 10	- - - 10 10	- - - 12 13	OR R ← R OR src			
TEST TESTB TESTL	dst	R IR DA X	7 8 11 12	- - 12 12	- - 14 15	13 13 16 17	- - 17 17	- - 19 20	Test dst OR 0
TCC TCCB	cc, dst	R	5	-	-	Test Condition Code Set LSB if cc is true			
XOR XORB	R, src	R IM IR DA X	4 7 7 9 10	- - - 10 10	- - - 12 13	Exclusive OR R ← R XOR src			

Program Control

Mnemonics	Operands	Addr. Modes	Clock Cycles						Operation
			Word, Byte			Long Word			
			NS	SS	SL	NS	SS	SL	
CALL	dst	IR	10	-	15				Call Subroutine Autodecrement SP @ SP ← PC PC ← dst
		DA	12	18	20				
		X	13	18	21				
CALR	dst	RA	10	-	15				Call Relative Autodecrement SP @ SP ← PC PC ← PC + dst (range -4094 to +4096)
DJNZ DBJNZ	R, dst	RA	11	-	-				Decrement and Jump if Non-Zero R ← R - 1 If R ≠ 0: PC ← PC + dst (range -254 to 0)
IRET*	-	-	13	-	16				Interrupt Return PS ← @ SP Autoincrement SP
JP	cc, dst	IR	10	-	15			(taken)	Jump Conditional If cc is true: PC ← dst
		IR	7	-	7			(not taken)	
		DA	7	8	10				
		X	8	8	11				
JR	cc, dst	RA	6	-	-				Jump Conditional Relative If cc is true: PC ← PC + dst (range -256 to +254)
RET	cc	-	10	-	13			(taken)	Return Conditional If cc is true: PC ← @ SP Autoincrement SP
			7	-	7			(not taken)	
SC	src	IM	33	-	39				System Call Autodecrement SP @ SP ← old PS Push instruction PS ← System Call PS

Bit Manipulation

BIT BITB	dst, b	R	4	-	-				Test Bit Static Z flag ← NOT dst bit specified by b
		IR	8	-	-				
		DA	10	11	13				
		X	11	11	14				
BIT BITB	dst, R	R	10	-	-				Test Bit Dynamic Z flag ← NOT dst bit specified by contents of R
RES RESB	dst, b	R	4	-	-				Reset Bit Static Reset dst bit specified by b
		IR	11	-	-				
		DA	13	14	16				
		X	14	14	17				
RES RESB	dst, R	R	10	-	-				Reset Bit Dynamic Reset dst bit specified by contents R
SET SETB	dst, b	R	4	-	-				Set Bit Static Set dst bit specified by b
		IR	11	-	-				
		DA	13	14	16				
		X	14	14	17				
SET SETB	dst, R	R	10	-	-				Set Bit Dynamic Set dst bit specified by contents of R
TSET TSETB	dst	R	7	-	-				Test and Set S flag ← MSB of dst dst ← all 1s
		IR	11	-	-				
		DA	14	15	17				
		X	15	15	18				

*Privileged instruction. Executed in system mode only.

Rotate and Shift	Mnemonics	Operands	Addr. Modes	Clock Cycles						Operation
				Word, Byte			Long Word			
				NS	SS	SL	NS	SS	SL	
	RLDB	R, src	R	9	-	-				Rotate Digit Left
	RRDB	R, src	R	9	-	-				Rotate Digit Right
	RL RLB	dst, n	R R	6 for n = 1 7 for n = 2						Rotate Left by n bits (n = 1, 2)
	RLC RLCB	dst, n	R R	6 for n = 1 7 for n = 2						Rotate Left through Carry by n bits (n = 1, 2)
	RR RRB	dst, n	R R	6 for n = 1 7 for n = 2						Rotate Right by n bits (n = 1, 2)
	RRC RRCB	dst, n	R R	6 for n = 1 7 for n = 2						Rotate Right through Carry by n bits (n = 1, 2)
	SDA SDAB SDAL	dst, R	R	(15 + 3 n)			(15 + 3 n)			Shift Dynamic Arithmetic Shift dst left or right by contents of R
	SDL SDLB SDLL	dst, R	R	(15 + 3 n)			(15 + 3 n)			Shift Dynamic Logical Shift dst left or right by contents of R
	SLA SLAB SLAL	dst, n	R	(13 + 3 n)			(13 + 3 n)			Shift Left Arithmetic by n bits
	SLL SLLB SLLL	dst, n	R	(13 + 3 n)			(13 + 3 n)			Shift Left Logical by n bits
	SRA SRAB SRAL	dst, n	R	(13 + 3 n)			(13 + 3 n)			Shift Right Arithmetic by n bits
	SRL SRLB SRL	dst, n	R	(13 + 3 n)			(13 + 3 n)			Shift Right Logical by n bits
Block Transfer and String Manipulation	CPD CPDB	R _X , src, R _Y , cc	IR	20	-	-				Compare and Decrement R _X ← src Autodecrement src address R _Y ← R _Y - 1
	CPDR CPDRB	R _X , src, R _Y , cc	IR	(11 + 9 n)						Compare, Decrement and Repeat R _X ← src Autodecrement src address R _Y ← R _Y - 1 Repeat until cc is true or R _Y = 0
	CPI CPIB	R _X , src, R _Y , cc	IR	20	-	-				Compare and Increment R _X ← src Autoincrement src address R _Y ← R _Y - 1
	CPIR CPIRB	R _X , src, R _Y , cc	IR	(11 + 9 n)						Compare, Increment and Repeat R _X ← src Autoincrement src address R _Y ← R _Y - 1 Repeat until cc is true or R _Y = 0
	CPSD CPSDB	dst, src, R, cc	IR	25	-	-				Compare String and Decrement dst ← src Autodecrement dst and src addresses R ← R - 1

Block Transfer and String Manipulation (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles						Operation
			Word. Byte			Long Word			
			NS	SS	SL	NS	SS	SL	
CPSDR CPSDRB	dst, src, R, cc	IR	(11 + 14 n)						Compare String, Decr. and Repeat dst ← src Autodecrement dst and src addresses R ← R - 1 Repeat until cc is true or R = 0
CPSI CPSIB	dst, src, R, cc	IR	25	-	-				Compare String and Increment dst ← src Autoincrement dst and src addresses R ← R - 1
CPSIR CPSIRB	dst, src, R, cc	IR	(11 + 14 n)						Compare String, Incr. and Repeat dst ← src Autoincrement dst and src addresses R ← R - 1 Repeat until cc is true or R = 0
LDD LDDB	dst, src, R	IR	20	-	-				Load and Decrement dst ← src Autodecrement dst and src addresses R ← R - 1
LDDR LDDRB	dst, src, R	IR	(11 + 9 n)						Load, Decrement and Repeat dst ← src Autodecrement dst and src addresses R ← R - 1 Repeat until R = 0
LDI LDIB	dst, src, R	IR	20	-	-				Load and Increment dst ← src Autoincrement dst and src addresses R ← R - 1
LDIR LDIRB	dst, src, R	IR	(11 + 9 n)						Load, Increment and Repeat dst ← src Autoincrement dst and src addresses R ← R - 1 Repeat until R = 0
TRDB	dst, src, R	IR	25	-	-				Translate and Decrement dst ← src (dst) Autodecrement dst address R ← R - 1
TRDRB	dst, src, R	IR	(11 + 14 n)						Translate, Decrement and Repeat dst ← src (dst) Autodecrement dst address R ← R - 1 Repeat until R = 0
TRIB	dst, src, R	IR	25	-	-				Translate and Increment dst ← src (dst) Autoincrement dst address R ← R - 1
TRIRB	dst, src, R	IR	(11 + 14 n)						Translate, Increment and Repeat dst ← src (dst) Autoincrement dst address R ← R - 1 Repeat until R = 0
TRTDB	src 1, src 2, R	IR	25	-	-				Translate and Test, Decrement RH1 ← src 2 (src 1) Autodecrement src 1 address R ← R - 1

2

**Block Transfer
and String
Manipulation**
(Continued)

Block Transfer and String Manipulation (Continued)	Mnemonics	Operands	Addr. Modes	Clock Cycles						Operation
				Word, Byte			Long Word			
				NS	SS	SL	NS	SS	SL	
	TRTDRB	src 1, src 2, R	IR	(11 + 14 n)						Translate and Test, Decr. and Repeat RH1 ← src 2 (src 1) Autodecrement src 1 address R ← R - 1 Repeat until R = 0 or RH1 = 0
	TRTIB	src 1, src 2, R	IR	25						Translate and Test, Increment RH1 ← src 2 (src 1) Autoincrement src 1 address R ← R - 1
	TRTIRB	src 1, src 2, R	IR	(11 + 14 n)						Translate and Test, Incr. and Repeat RH1 ← src 2 (src 1) Autoincrement src 1 address R ← R - 1 Repeat until R = 0 or RH1 = 0
Input/ Output	IN*	R, src	IR	10	-	-				Input R ← src
	INB*		DA	12	-	-				
	IND*	dst, src, R	IR	21	-	-				Input and Decrement dst ← src Autodecrement dst address R ← R - 1
	INDB*									
	INDR*	dst, src, R	IR	(11 + 10 n)						Input, Decrement and Repeat dst ← src Autodecrement dst address R ← R - 1 Repeat until R = 0
	INDRB*									
	INI*	dst, src, R	IR	21	-	-				Input and Increment dst ← src Autoincrement dst address R ← R - 1
	INIB*									
	INIR*	dst, src, R	IR	(11 + 10 n)						Input, Increment and Repeat dst ← src Autoincrement dst address R ← R - 1 Repeat until R = 0
	INIRB*									
	OUT*	dst, R	IR	10	-	-				Output dst ← R
	OUTB*		DA	12	-	-				
OUTD*	dst, src, R	IR	21	-	-				Output and Decrement dst ← src Autodecrement src address R ← R - 1	
OUTDB*										
OTDR*	dst, src, R	IR	(11 + 10 n)						Output, Decrement and Repeat dst ← src Autodecrement src address R ← R - 1 Repeat until R = 0	
OTDRB*										
OUTI*	dst, src, R	IR	21	-	-				Output and Increment dst ← src Autoincrement src address R ← R - 1	
OUTIB*										
OTIR*	dst, src, R	IR	(11 + 10 n)						Output, Increment and Repeat dst ← src Autoincrement src address R ← R - 1 Repeat until R = 0	
OTIRB*										

*Privileged instructions. Executed in system mode only.

Input/Output (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles						Operation
			Word, Byte			Long Word			
NS	SS	SL	NS	SS	SL				
SIN* SINB*	R, src	DA	12	-	-				Special Input R ← src
SIND* SINDB*	dst, src, R	IR	21	-	-				Special Input and Decrement dst ← src Autodecrement dst address R ← R - 1
SINDR* SINDRB*	dst, src, R	IR	(11 + 10 n)						Special Input, Decrement and Repeat dst ← src Autodecrement dst address R ← R - 1 Repeat until R = 0
SINI* SINIB*	dst, src, R	IR	21	-	-				Special Input and Increment dst ← src Autoincrement dst address R ← R - 1
SINIR* SINIRB*	dst, src, R	IR	(11 + 10 n)						Special Input, Increment and Repeat dst ← src Autoincrement dst address R ← R - 1 Repeat until R = 0
SOUT* SOUTB*	dst, src	DA	12	-	-				Special Output dst ← src
SOUTD* SOUTDB*	dst, src, R	IR	21	-	-				Special Output and Decrement dst ← src Autodecrement src address R ← R - 1
SOTDR* SOTDRB*	dst, src, R	IR	(11 + 10 n)						Special Output, Decr. and Repeat dst ← src Autodecrement src address R ← R - 1 Repeat until R = 0
SOUTI* SOUTIB*	dst, src, R	IR	21	-	-				Special Output and Increment dst ← src Autoincrement src address R ← R - 1
SOTIR* SOTIRB*	dst, src, R	R	(11 + 10 n)						Special Output, Incr. and Repeat dst ← src Autoincrement src address R ← R - 1 Repeat until R = 0
COMFLG	flags	-	7	-	-				Complement Flag (Any combination of C, Z, S, P/V)
DI*	int	-	6	-	-				Disable Interrupt (Any combination of NVI, VI)
EI*	int	-	6	-	-				Enable Interrupt (Any combination of NVI, VI)
HALT*	-	-	(8 + 3 n)						HALT
LDCTL*	CTLR, src	R	7	-	-				Load into Control Register CTLR ← src
LDCTL*	dst, CTLR	R	7	-	-				Load from Control Register dst ← CTLR

*Privileged instructions. Executed in system mode only.

CPU Control (Continued)	Mnemonics	Operands	Addr. Modes	Clock Cycles						Operation
				Word, Byte			Long Word			
				NS	SS	SL	NS	SS	SL	
	LDCTLB	FLGR, src	R	7	-	-				Load into Flag Byte Register FLGR ← src
	LDCTLB	dst, FLGR	R	7	-	-				Load from Flag Byte Register dst ← FLGR
	LDPS*	src	IR DA X	12 16 17	- 20 20	16 22 23				Load Program Status PS ← src
	MBIT*	-	-	7	-	-				Test Multi-Micro Bit Set S if $\overline{M}_1$ is Low; reset S if $\overline{M}_1$ is High.
	MREQ*	dst	R	(12 + 7 n)						Multi-Micro Request
	MRES*	-	-	5	-	-				Multi-Micro Reset
	MSET*	-	-	5	-	-				Multi-Micro Set
	NOP	-	-	7	-	-				No Operation
	RESFLG	flag	-	7	-	-				Reset Flag (Any combination of C, Z, S, P/V)
	SETFLG	flag	-	7	-	-				Set Flag (Any combination of C, Z, S, P/V)
*Privileged instructions. Executed in system mode only.										

*Privileged instructions. Executed in system mode only.

Condition Codes	Code	Meaning	Flag Settings	CC Field
		Always false	-	0000
		Always true	-	1000
	Z	Zero	Z = 1	0110
	NZ	Not zero	Z = 0	1110
	C	Carry	C = 1	0111
	NC	No Carry	C = 0	1111
	PL	Plus	S = 0	1101
	MI	Minus	S = 1	0101
	NE	Not equal	Z = 0	1110
	EQ	Equal	Z = 1	0110
	OV	Overflow	P/V = 1	0100
	NOV	No overflow	P/V = 0	1100
	PE	Parity is even	P/V = 1	0100
	PO	Parity is odd	P/V = 0	1100
	GE	Greater than or equal (signed)	(S XOR P/V) = 0	1001
	LT	Less than (signed)	(S XOR P/V) = 1	0001
	GT	Greater than (signed)	[Z OR (S XOR P/V)] = 0	1010
	LE	Less than or equal (signed)	[Z OR (S XOR P/V)] = 1	0010
	UGE	Unsigned greater than or equal	C = 0	1111
	ULT	Unsigned less than	C = 1	0111
	UGT	Unsigned greater than	[(C = 0) AND (Z = 0)] = 1	1011
	ULE	Unsigned less than or equal	(C OR Z) = 1	0011

Note that some condition codes have identical flag settings and binary fields in the instruction:
Z = EQ, NZ = NE, C = ULT, NC = UGE, OV = PE, NOV = PO

Status Line Codes	ST ₃ -ST ₀	Definition	ST ₃ -ST ₀	Definition
	0 0 0 0	Internal operation	1 0 0 0	Data memory request
	0 0 0 1	Memory refresh	1 0 0 1	Stack memory request
	0 0 1 0	I/O reference	1 0 1 0	Data memory request (EPU)
	0 0 1 1	Special I/O reference (e.g., to an MMU)	1 0 1 1	Stack memory request (EPU)
	0 1 0 0	Segment trap acknowledge	1 1 0 0	Program reference, nth word
	0 1 0 1	Non-maskable interrupt acknowledge	1 1 0 1	Instruction fetch, first word
	0 1 1 0	Non-vectored interrupt acknowledge	1 1 1 0	Extension processor transfer
	0 1 1 1	Vectored interrupt acknowledge	1 1 1 1	Reserved

**Pin
Description**

AD₀-AD₁₅. *Address/Data* (inputs/outputs, active High, 3-state). These multiplexed address and data lines are used both for I/O and to address memory.

AS. *Address Strobe* (output, active Low, 3-state). The rising edge of $\overline{AS}$ indicates addresses are valid.

BUSAK. *Bus Acknowledge* (output, active Low). A Low on this line indicates the CPU has relinquished control of the bus.

BUSRQ. *Bus Request* (input, active Low). This line must be driven Low to request the bus from the CPU.

DS. *Data Strobe* (output, active Low, 3-state). This line times the data in and out of the CPU.

MREQ. *Memory Request* (output, active Low, 3-state). A Low on this line indicates that the address/data bus hold a memory address.

M_I, M_O. *Multi-Micro In, Multi-Micro Out* (input and output, active Low). These two lines form a resource-request daisy chain that allows one CPU in a multi-microprocessor system to access a shared resource.

NMI. *Non-Maskable Interrupt* (edge triggered, input, active Low). A high-to-low transition on NMI requests a non-maskable interrupt. The NMI interrupt has the highest priority of the three types of interrupts.

NVI. *Non-Vectored Interrupt* (input, active Low). A Low on this line requests a non-vectored interrupt.

CLK. *System Clock* (input). CLK is a 5V single-phase time-base input.

RESET. *Reset* (input, active Low). A Low on this line resets the CPU.

R/W. *Read/Write* (output, Low = Write, 3-state). R/W indicates that the CPU is reading from or writing to memory or I/O.

SN₀-SN₆. *Segment Number* (outputs, active High, 3-state). These lines provide the 7-bit segment number used to address one of 128 segments by the Z8010 Memory Management Unit. Output by the Z8001 only.

SEGT. *Segment Trap* (input, active Low). The Memory Management Unit interrupts the CPU with a Low on this line when the MMU detects a segmentation trap.

ST₀-ST₃. *Status* (outputs, active High, 3-state). These lines specify the CPU status (see table).

STOP. *Stop* (input, active Low). This input can be used to single-step instruction execution.

VI. *Vectored Interrupt* (input, active Low). A Low on this line requests a vectored interrupt.

WAIT. *Wait* (input, active Low). This line indicates to the CPU that the memory or I/O device is not ready for data transfer.

B/W. *Byte/Word* (output, Low = Word, 3-state). This signal defines the type of memory reference on the 16-bit address/data bus.

N/S. *Normal/System Mode* (output, Low = System Mode, 3-state). N/S indicates the CPU is in the normal or system mode.

Reserved. Do not connect.

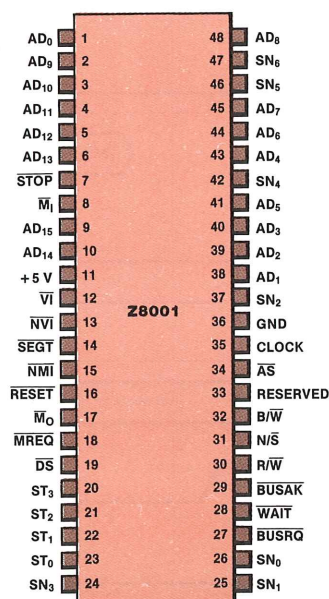


Figure 8. Z8001 Pin Assignments

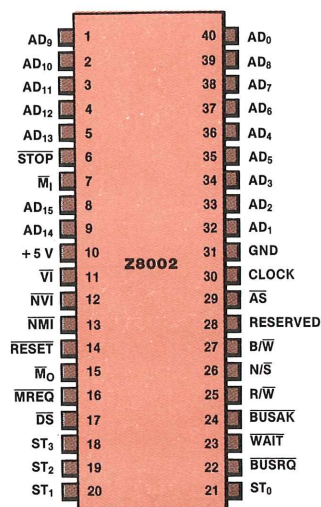


Figure 9. Z8002 Pin Assignments

Z8000 CPU Timing

The Z8000 CPU executes instructions by stepping through sequences of basic machine cycles, such as memory read or write, I/O device read or write, interrupt acknowledge and internal execution. Each of these basic cycles require three to ten clock cycles to execute. Instructions that require more clock cycles to execute are broken up into several machine cycles. Thus no machine cycle is longer than ten clock cycles and fast response to a Bus Request is guaranteed.

The instruction opcode is fetched by a normal memory read operation. A memory refresh cycle can be inserted just after the completion of any first instruction fetch (IF₁)

cycle and can also be inserted while the following instructions are being executed: MULT, MULTL, DIV, DIVL, HALT, all Shift instructions, all Block Move instructions, and the Multi-Micro Request instruction (MREQ).

The following timing diagrams show the relative timing relationships of all CPU signals during each of the basic operations. When a machine cycle requires additional clock cycles for CPU internal operation, one to five clock cycles are added. Memory and I/O read and write, as well as interrupt acknowledge cycles, can be extended by activating the $\overline{\text{WAIT}}$ input. For exact timing information, refer to the composite timing diagram.

Memory Read and Write

Memory read and instruction fetch cycles are identical, except for the status information on the ST₀-ST₃ outputs. During a memory read cycle, a 16-bit address is placed on the AD₀-AD₁₅ outputs early in the first clock

period, as shown in Figure 10. (In the Z8001, the 7-bit segment number is output on SN₀-SN₆ one clock period earlier than the 16-bit address offset to compensate for the delay in the memory management circuitry.)

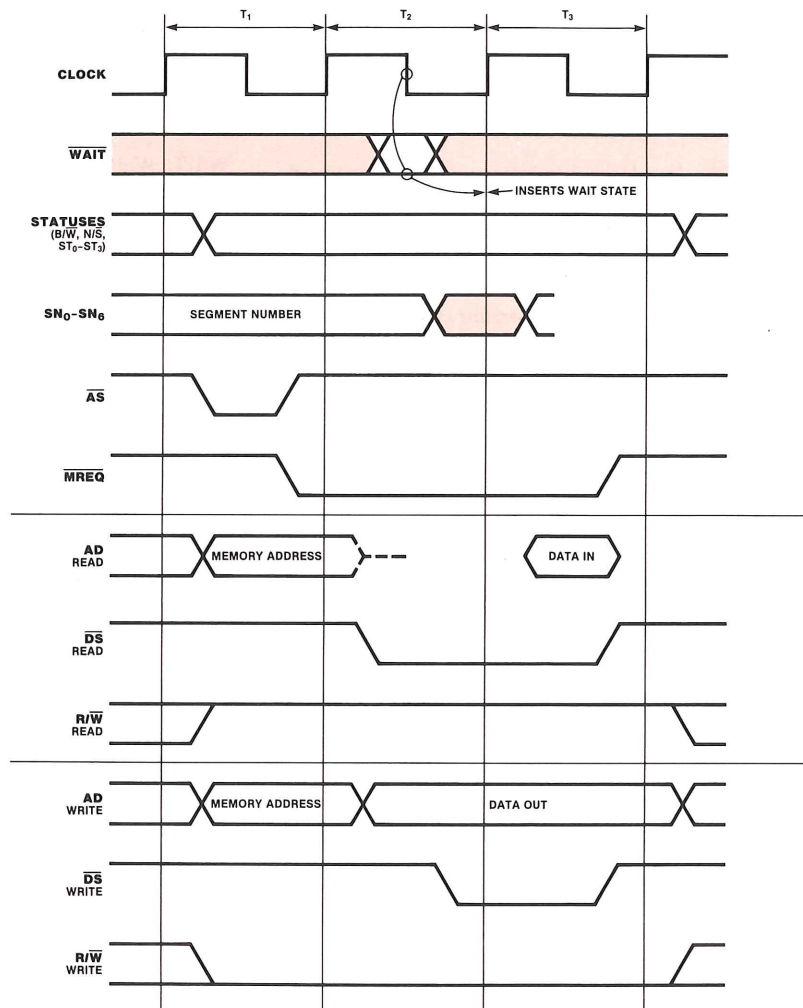


Figure 10. Memory Read and Write Timing

Memory Read and Write (Continued)

A valid address is indicated by the rising edge of Address Strobe. Status and mode information become valid early in the memory access cycle and remain stable throughout. The state of the **WAIT** input is sampled in the middle of the second clock cycle by the falling edge of **Clock**. If **WAIT** is Low, an additional clock period is added between T_2 and T_3 . **WAIT** is sampled again in the middle of this wait cycle, and additional wait states can be inserted. This allows interfacing slow

memories. No control outputs change during wait states.

Although Z8000 memory is word organized, memory is addressed as bytes. All instructions are word-aligned, using even addresses. Within a 16-bit word, the most significant byte (D_8-D_{15}) is addressed by the low-order address ($A_0 = \text{Low}$), and the least significant byte (D_0-D_7) is addressed by the high-order address ($A_0 = \text{High}$).

Input/ Output

I/O timing is similar to memory read/write timing, except that one wait state is automatically inserted between T_2 and T_3 (Figure 11).

Both the segmented Z8001 and the non-segmented Z8002 use 16-bit I/O addresses.

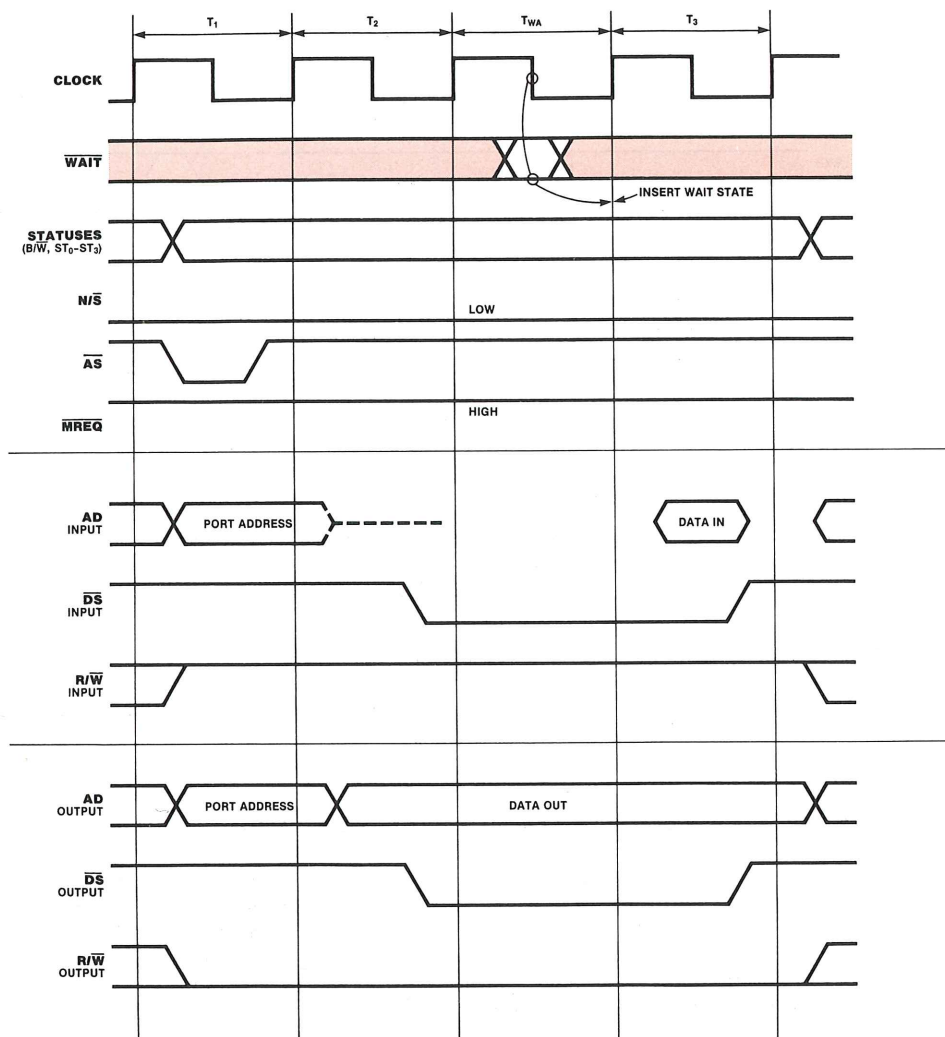


Figure 11. Input/Output Timing

Interrupt and Segment Trap Request and Acknowledge

The Z8000 CPU recognizes three interrupt inputs (non-maskable, vectored and non-vectored) and a segmentation trap input. Any High-to-Low transition on the $\overline{\text{NMI}}$ input is asynchronously edge detected and sets the internal NMI latch. The $\overline{\text{VI}}$, $\overline{\text{NVI}}$ and $\overline{\text{SEGT}}$ inputs as well as the state of the internal NMI latch are sampled at the beginning of T_3 in the last machine cycle of any instruction.

In response to an interrupt or trap, the subsequent IF_1 cycle is exercised, but aborted. The program counter is not updated, but the system stack pointer is decremented.

The next machine cycle is the interrupt acknowledge cycle. This cycle has five automatic wait states, with additional wait

states possible, as shown in Figure 12.

After the last wait state, the CPU reads the information on AD_{0-15} and stores it temporarily, to be saved on the stack later in the acknowledge sequence. This word identifies the source of the interrupt or trap. For the non-vectored and non-maskable interrupts, all 16 bits can represent peripheral device status information. For the vectored interrupt, the low byte is the jump vector, and the high byte can be extra user status. For the segmentation trap, the *high* byte is the Memory Management Unit identifier and the *low* byte is undefined.

After the acknowledge cycle, the N/S output indicates the automatic change to system mode.

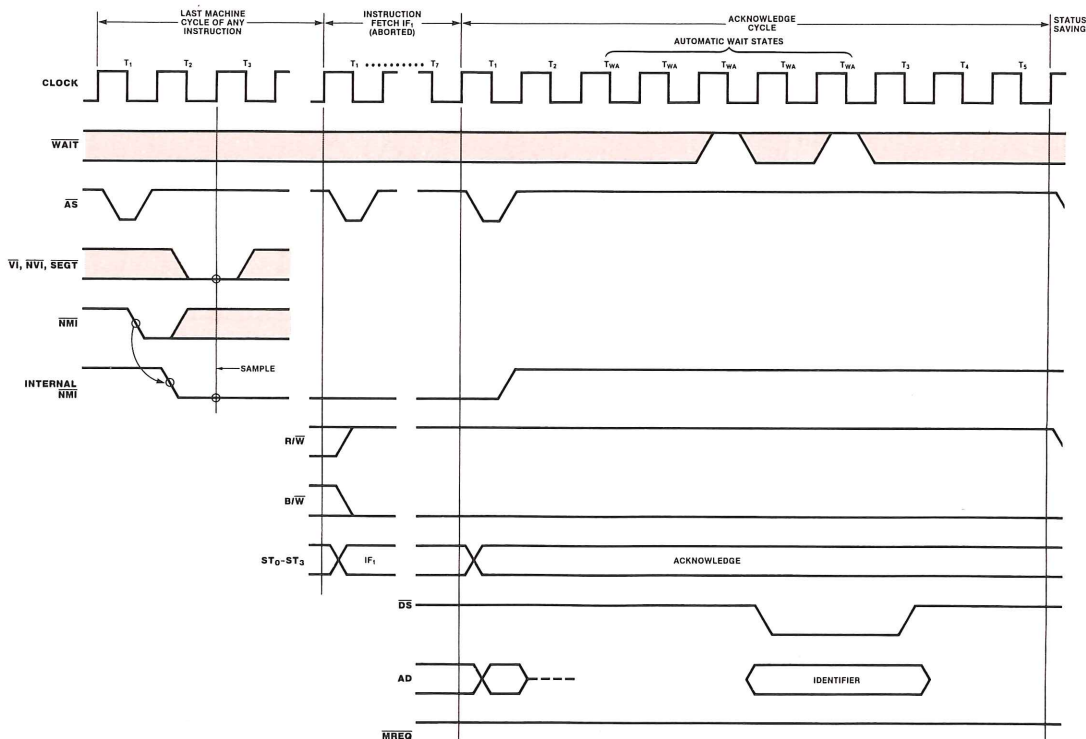


Figure 12. Interrupt and Segment Trap Request/Acknowledge Timing

Status Saving Sequence

The machine cycles following the interrupt acknowledge or segmentation trap acknowledge cycle push the old status information on the system stack in the following order: the 16-bit program counter; the 7-bit segment number (Z8001 only); the flag and control

word; and finally the interrupt/trap identifier. Subsequent machine cycles fetch the new program status from the program status area, and then branch to the interrupt/trap service routine.

Bus Request Acknowledge Timing

A Low on the $\overline{\text{BUSRQ}}$ input indicates to the CPU that another device is requesting the Address/Data and Control buses. The asynchronous $\overline{\text{BUSRQ}}$ input is synchronized at the beginning of any machine cycle (Figure 13). If

$\overline{\text{BUSRQ}}$ is Low, an internal synchronous $\overline{\text{BUSRQ}}$ signal is generated, which—after completion of the current machine cycle—causes the $\overline{\text{BUSAK}}$ output to go Low and all bus outputs to go into the high-impedance state. The

Bus Request/ Acknowledge (Continued)

requesting device—typically a DMA—can then control all buses.

When $\overline{\text{BUSRQ}}$ is released, it is synchronized with the rising clock edge and the $\overline{\text{BUSAK}}$ out-

put goes High one clock period later, indicating that the CPU will again take control of the bus.

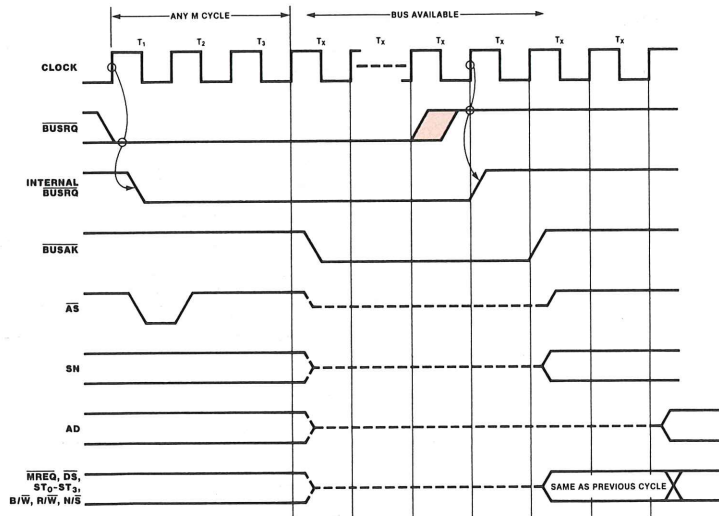


Figure 13. Bus Request/Acknowledge Timing

Stop

The $\overline{\text{STOP}}$ input is sampled by the last falling clock edge immediately preceding any IF_1 cycle (Figure 14). If $\overline{\text{STOP}}$ is found Low, a stream of memory refresh cycles is inserted after T_3 , again sampling the $\overline{\text{STOP}}$ input on each falling clock edge in the middle of the T_3 states. This refresh operation does not use the

refresh prescaler or its divide-by-four clock prescaler; rather, it double-increments the refresh counter every three clock cycles. When $\overline{\text{STOP}}$ is found High again, the next refresh cycle is completed, any remaining T states of the IF_1 cycle are then executed and the CPU continues its operation.

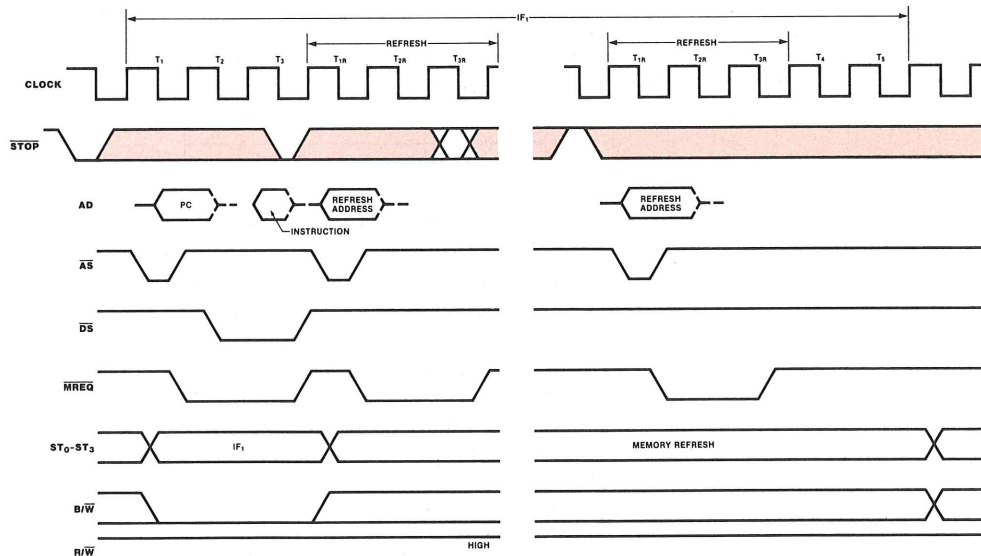


Figure 14. Stop Timing

Internal Operation

Certain extended instructions such as Multiply and Divide, and some special instructions need additional time for the execution of internal operations. In these cases, the CPU goes through a sequence of internal operation machine cycles, each of which is three to eight clock cycles long (Figure 15). This allows fast response to Bus Request and Refresh Request,

because bus request or refresh cycles can be inserted at the end of any internal machine cycle.

Although the address outputs during T_1 are undefined, Address Strobe is generated to satisfy the requirements of future Z-Bus compatible self-refresh dynamic memories.

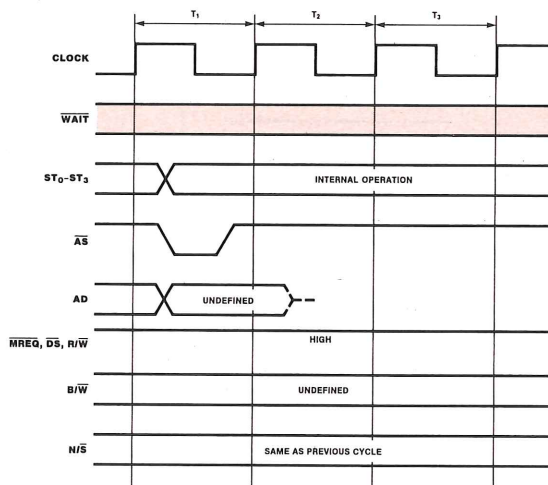


Figure 15. Internal Operation Timing

Memory Refresh

When the 6-bit prescaler in the refresh counter has been decremented to zero, a refresh cycle consisting of three T-states is started as soon as possible (that is, after the next IF_1 cycle or Internal Operation cycle).

The 9-bit refresh counter value is put on the low-order side of the address bus (AD_0 - AD_8); AD_9 - AD_{15} are undefined (Figure 16). Since the memory is word-organized, A_0 is always Low during refresh and the refresh counter is

always incremented by two, thus stepping through 256 consecutive refresh addresses on AD_1 - AD_8 . Unless disabled, the presetable prescaler runs continuously and the delay in starting a refresh cycle is therefore not cumulative.

While the $\overline{STOP}$ input is Low, a continuous stream of memory refresh cycles, each three T-states long, is executed without using the refresh prescaler.

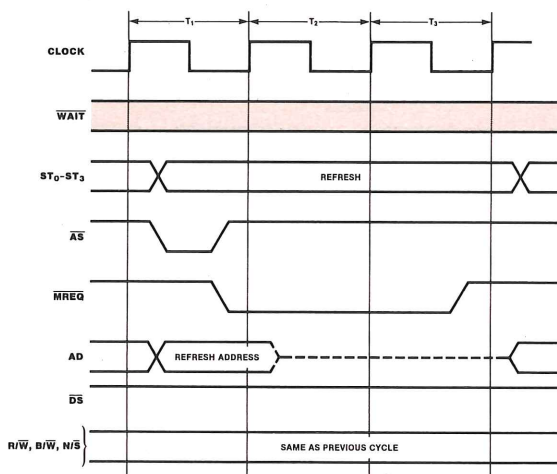


Figure 16. Memory Refresh Timing

Halt

A HALT instruction executes an unlimited number of 3-cycle internal operations, inter-spersed with memory refresh cycles whenever requested. An interrupt, segmentation trap or reset are the only exits from a HALT instruction.

Reset

A Low on the RESET input causes the following results within five clock cycles or less (Figure 17):

- AD_0-AD_{15} are 3-stated
- $\overline{AS}$, $\overline{DS}$, $\overline{MREQ}$, $\overline{BUSA\overline{K}}$ and $\overline{M\overline{O}}$ are forced High
- ST_0-ST_3 and SN_0-SN_6 are forced Low
- Refresh is disabled
- $R/\overline{W}$, $B/\overline{W}$ and $N/\overline{S}$ are not affected

When RESET has been High for three clock

The CPU samples the $\overline{VI}$, $\overline{NVI}$, $\overline{NMI}$ and $\overline{SEGT}$ inputs at the beginning of every T_3 cycle. If an input is found active during two consecutive samples, the subsequent IF_1 cycle is exercised, but aborted, and the normal interrupt acknowledge cycle is started.

periods, two consecutive memory read cycles are executed in the system mode. In the Z8001, the first cycle reads the flag and control word from location 0002, the next reads the 7-bit program counter segment number from location 0004, the next reads the 16-bit PC offset from location 0006, and the following IF_1 cycle starts the program. In the Z8002, the first cycle reads the flag and control word from location 0002, the next reads the PC from location 0004 and the following IF_1 cycle starts the program.

2

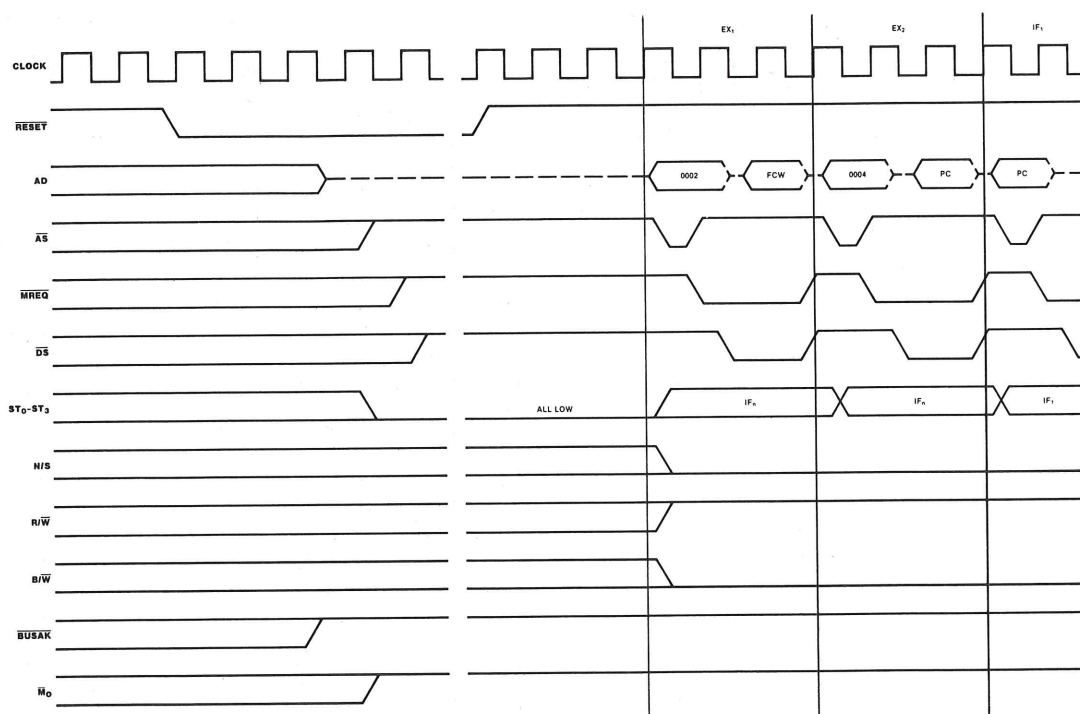
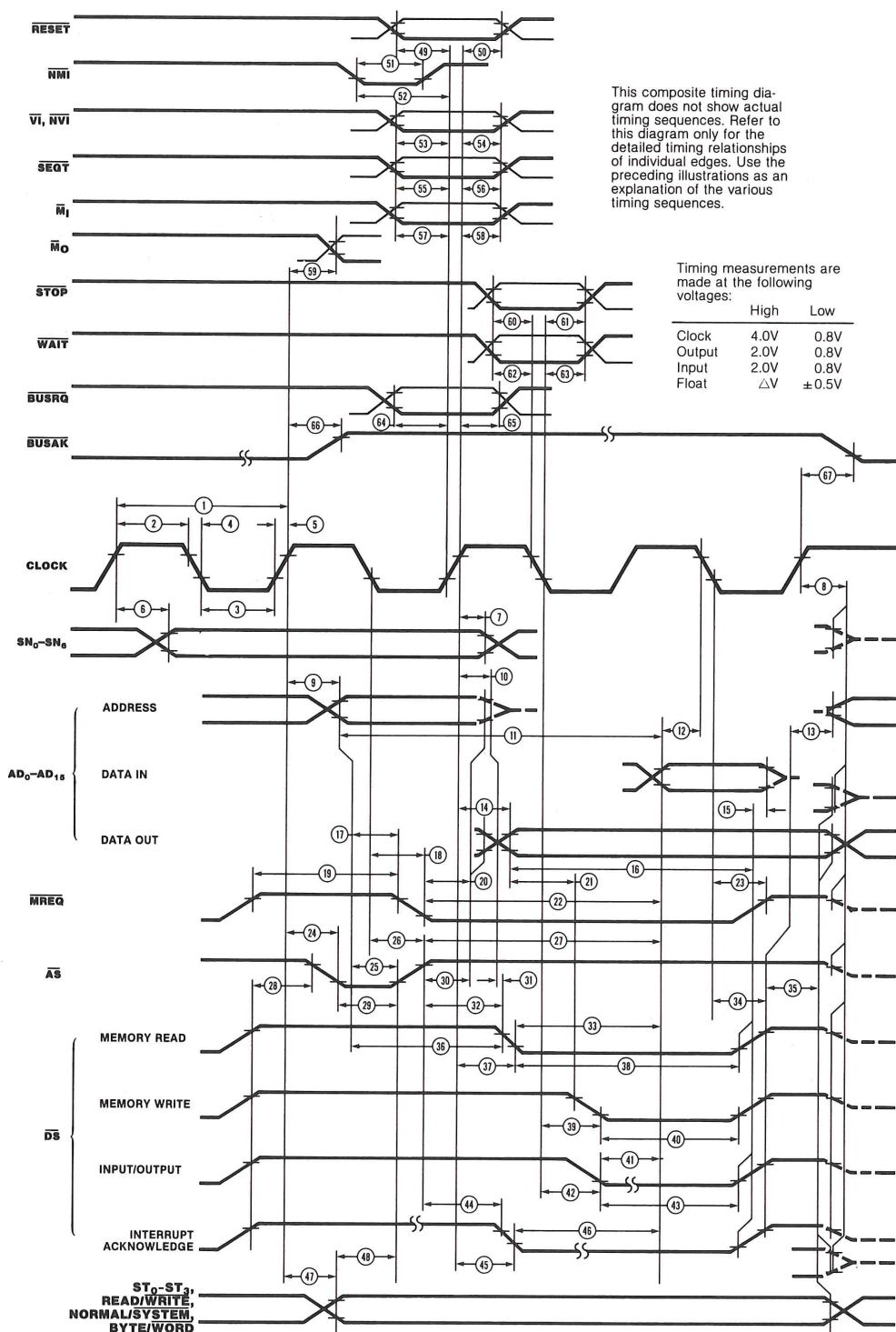


Figure 17. Reset Timing

Composite AC Timing Diagram



AC	Number	Symbol	Parameter	Min	Max	Unit
Electrical Characteristics	1	T _{cC}	Clock Cycle Time	250	2000	ns
	2	T _{wCh}	Clock Width (High)	105	2000	ns
	3	T _{wCl}	Clock Width (Low)	105	2000	ns
	4	T _{fC}	Clock Fall Time		20	ns
	5	T _{rC}	Clock Rise Time		20	ns
	6	T _{dC} (SN _v)	Clock ↑ to Segment Number Valid (50 pF load)		130	ns
	7	T _{dC} (SN _n)	Clock ↑ to Segment Number Not Valid	20		ns
	8	T _{dC} (B _z)	Clock ↑ to Bus Float		65	ns
	9	T _{dC} (A)	Clock ↑ to Address Valid		100	ns
	10	T _{dC} (A _z)	Clock ↑ to Address Float		65	ns
	11	T _{dA} (DI)	Address Valid to Data In Required Valid	400		ns
	12	T _{sDI} (C)	Data In to Clock ↑ Setup Time	50		ns
	13	T _{dDS} (A)	$\overline{DS}$ ↑ to Address Active	80		ns
	14	T _{dC} (DO)	Clock ↑ to Data Out Valid		100	ns
	15	T _{hDI} (DS)	Data In to $\overline{DS}$ ↑ Hold Time	0		ns
	16	T _{dDO} (DS)	Data Out Valid to $\overline{DS}$ ↑ Delay	230		ns
	17	T _{dA} (MR)	Address Valid to $\overline{MREQ}$ ↓ Delay	55		ns
	18	T _{dC} (MR)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		80	ns
	19	T _{wMRh}	$\overline{MREQ}$ Width (High)	190		ns
	20	T _{dMR} (A)	$\overline{MREQ}$ ↓ to Address Not Active	70		ns
	21	T _{dDO} (DSW)	Data Out Valid to $\overline{DS}$ ↓ (Write) Delay	55		ns
	22	T _{dMR} (DI)	$\overline{MREQ}$ ↓ to Data In Required Valid	330		ns
	23	T _{dC} (MR)	Clock ↓ to $\overline{MREQ}$ ↑ Delay		80	ns
	24	T _{dC} (AS _f)	Clock ↑ to $\overline{AS}$ ↓ Delay		80	ns
	25	T _{dA} (AS)	Address Valid to $\overline{AS}$ ↑ Delay	55		ns
	26	T _{dC} (AS _r)	Clock ↓ to $\overline{AS}$ ↑ Delay		90	ns
	27	T _{dAS} (DI)	$\overline{AS}$ ↑ to Data In Required Valid	290		ns
	28	T _{dDS} (AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		ns
	29	T _{wAS}	$\overline{AS}$ Width (Low)	80		ns
	30	T _{dAS} (A)	$\overline{AS}$ ↑ to Address Not Active Delay	60		ns
	31	T _{dAz} (DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		ns
	32	T _{dAS} (DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ (Read) ↓ Delay	70		ns
	33	T _{dDSR} (DI)	$\overline{DS}$ (Read) ↓ to Data In Required Valid	155		ns
	34	T _{dC} (DS _r)	Clock ↓ to $\overline{DS}$ ↑ Delay		70	ns
	35	T _{dDS} (DO)	$\overline{DS}$ ↑ to Data Out and STATUS Not Valid	80		ns
	36	T _{dA} (DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	120		ns
	37	T _{dC} (DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		120	ns
	38	T _{wDSR}	$\overline{DS}$ (Read) Width (Low)	275		ns
	39	T _{dC} (DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		95	ns
	40	T _{wDSW}	$\overline{DS}$ (Write) Width (Low)	160		ns
	41	T _{dDSI} (DI)	$\overline{DS}$ (Input) ↓ to Data In Required Valid	315		ns
	42	T _{dC} (DS _f)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		120	ns
	43	T _{wDS}	$\overline{DS}$ (I/O) Width (Low)	400		ns
	44	T _{dAS} (DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	960		ns
	45	T _{dC} (DSA)	Clock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		120	ns
	46	T _{dDSA} (DI)	$\overline{DS}$ (Ack.) ↓ to Data In Required Delay	420		ns
	47	T _{dC} (S)	Clock ↑ to Status Valid Delay		110	ns
	48	T _{dS} (AS)	Status Valid to $\overline{AS}$ ↑ Delay	60		ns
	49	T _{sR} (C)	$\overline{RESET}$ to Clock ↑ Setup Time	180		ns
	50	T _{hR} (C)	$\overline{RESET}$ to Clock ↑ Hold Time	0		ns
	51	T _{wNMI}	$\overline{NMI}$ Width (Low)	100		ns
	52	T _{sNMI} (C)	$\overline{NMI}$ to Clock ↑ Setup Time	140		ns
	53	T _{sVI} (C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Setup Time	110		ns
	54	T _{hVI} (C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Hold Time	0		ns
	55	T _{sSGT} (C)	$\overline{SEGT}$ to Clock ↑ Setup Time	70		ns
	56	T _{hSGT} (C)	$\overline{SEGT}$ to Clock ↑ Hold Time	0		ns
	57	T _{sMi} (C)	$\overline{MI}$ to Clock ↑ Setup Time	180		ns
	58	T _{hMi} (C)	$\overline{MI}$ to Clock ↑ Hold Time	0		ns
	59	T _{dC} (Mo)	Clock ↑ to $\overline{MO}$ Delay		120	ns
	60	T _{sSTP} (C)	$\overline{STOP}$ to Clock ↓ Setup Time	140		ns
	61	T _{hSTP} (C)	$\overline{STOP}$ to Clock ↓ Hold Time	0		ns
	62	T _{sWT} (C)	$\overline{WAIT}$ to Clock ↓ Setup Time	50		ns
	63	T _{hWT} (C)	$\overline{WAIT}$ to Clock ↓ Hold Time	0		ns
	64	T _{sBRQ} (C)	$\overline{BUSERQ}$ to Clock ↑ Setup Time	90		ns
	65	T _{hBRQ} (C)	$\overline{BUSERQ}$ to Clock ↑ Hold Time	0		ns
	66	T _{dC} (BAK _r)	Clock ↑ to $\overline{BUSAK}$ ↑ Delay		100	ns
	67	T _{dC} (BAK _f)	Clock ↑ to $\overline{BUSAK}$ ↓ Delay		100	ns

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND -0.3 V to +7.0 V

Operating Ambient Temperature 0°C to +70°C

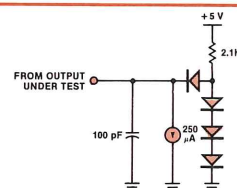
Storage Temperature -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\text{ V} \leq V_{CC} \leq +5.25\text{ V}$
- $GND = 0\text{ V}$
- $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$



All ac parameters assume a load capacitance of 100 pF max, except for parameter 6 (50 pF max). Timing references between two output signals assume a load difference of 50 pF max.

DC Characteristics	Symbol	Parameter	Min	Max	Unit	Condition
	V_{CH}	Clock Input High Voltage	$V_{CC}-0.4$	$V_{CC}+0.3$	V	Driven by External Clock Generator
	V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by External Clock Generator
	V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
	V_{IL}	Input Low Voltage	-0.3	0.8	V	
	V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\text{ }\mu\text{A}$
	V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0\text{ mA}$
	I_{IL}	Input Leakage		± 10	μA	$0.4 \leq V_{IN} \leq +2.4\text{ V}$
	I_{OL}	Output Leakage		± 10	μA	$0.4 \leq V_{OUT} \leq +2.4\text{ V}$
	I_{CC}	V_{CC} Supply Current		300	mA	

Advanced Architectural Features of the Z8000 CPU



NEW
1980

Tutorial Information

February 1980

Introduction

The Zilog Z8000 CPU microprocessor is a major advance in microcomputer architecture. It offers many minicomputer and mainframe features for the first time in a microprocessor chip. This tutorial describes the Z8000 CPU with emphasis placed on those features that set it apart from its microprocessor predecessors. For a detailed description of all Z8000 CPU features, consult the Zilog publications listed in the bibliography at the end of this tutorial.

The features to be discussed are grouped into four areas: CPU organization, handling of interrupts and traps, use of memory, and new

instructions and data capabilities.

Before discussing these features in more detail, a word about nomenclature is in order. The term Z8000 refers to the concept and architecture of a family of parts. Zilog has adopted the typical semiconductor industry 4-digit designation for Z8000 Family parts, while also keeping the traditional 3-letter acronym that proved so popular for the Z-80 Family. Thus, the 48-pin version of the Z8000 CPU is called the Z8001 CPU; the 40-pin version is known as the Z8002 CPU.

2

CPU

Organization

The Z8000 CPU is organized around a general-purpose register file (Figure 1). The register file is a group of registers, any one of which can be used as an accumulator, index register, memory pointer, stack pointer, etc. The only exception is Register 0, as explained later.

Flexibility is the major advantage of a general-purpose register organization over an organization that dedicates particular registers to each function. Computation-oriented routines can use general registers as accumulators for intermediate results whereas data manipulation routines can use these registers for memory pointers.

Dedicated registers, however, have a disadvantage: when more registers of a given type are needed than are supplied by the machine, the performance degrades by the extra instructions to swap registers and memory locations. For example, a processor with two index registers suffers when three are needed because a temporary variable in memory (or in another register) must be used for the third

index. When the third index is needed, it must be swapped into an index register. In contrast, on a general-register machine three of the registers could be dedicated for index use. In addition, since the need for index registers may vary over the course of a program, a general-register architecture, such as the Z8000, can be adapted to the changing needs of the computation with respect to the number of accumulators, memory pointers and index registers. Thus flexibility results in increased performance and ease of use.

In addition, the registers of the Z8000 are organized to process 8-bit bytes, 16-bit words, 32-bit long words and 64-bit quadruple words. This readily accommodates applications that process data of variable sizes as well as different tasks that require different data sizes.

Although all registers can—in general—be used for any purpose, certain instructions such as Subroutine Call and String Translation make use of specific registers in the general register file, and this must be taken into account when these instructions are used.

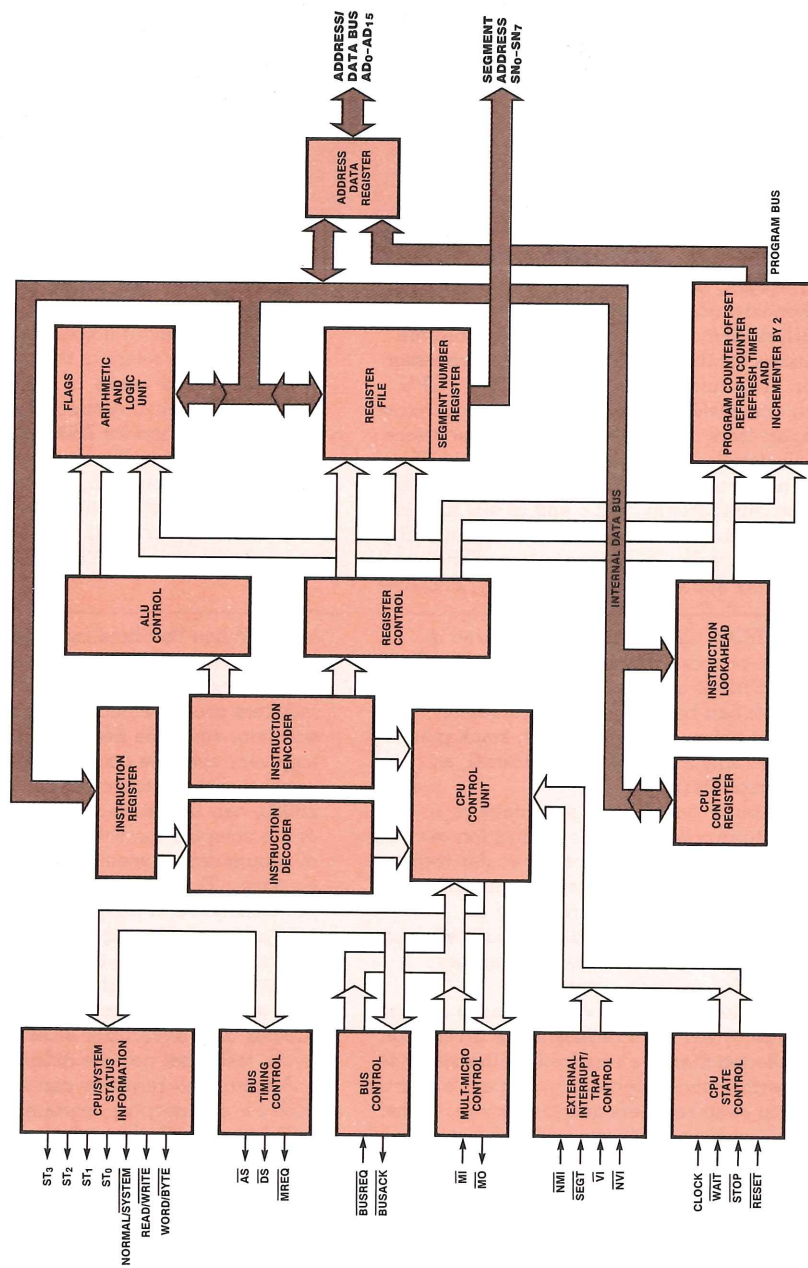


Figure 1. Instruction Register

The Z8000 CPU also contains a number of special-purpose registers in addition to the general-purpose ones. These include the Program Counter, Program Status registers and

the Refresh Counter. These registers are accessible through software and provide some of the interesting features of Z8000 CPU architecture.

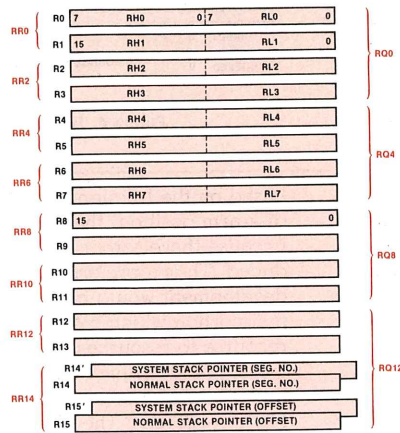


Figure 2. Z8001 General Purpose Registers

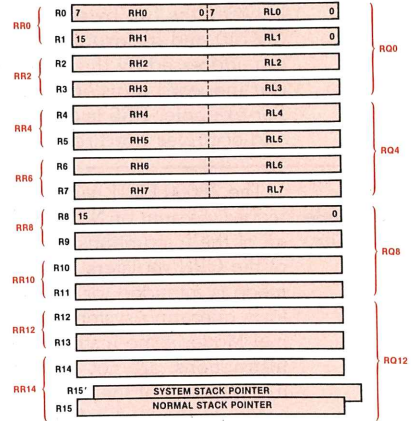


Figure 3. Z8002 General Purpose Registers

**Register
Organization**

All general-purpose registers can be used as accumulators, and all but one as index registers or memory pointers. The one register that cannot be used as an index register is Register 0. Specifying Register 0 is used as an escape mechanism to change the address mode from IR to IM, from X to DA, or—with Load instructions—from BA to RA. This has been done so that the two addressing mode bits in the instruction can specify more than four addressing modes for the same opcode.

The Z8000 CPU register file can be addressed in several groupings: as sixteen byte registers (occupying the upper half of the file only), as sixteen word registers, as eight long-word registers, as four quadruple-word registers, or as a mixture of these. Instructions either explicitly or implicitly specify the type of register. Table 1 illustrates the correspondence between the 4-bit source and destination register fields in the instruction (Figure 4) and the location of the registers in the register file (Figures 2 and 3).

Register Designator	Byte	Word	Long Word	Quadruple Word
0 0 0 0	RH0	R0	RR0	RQ0
0 0 0 1	RH1	R1		
0 0 1 0	RH2	R2	RR2	
0 0 1 1	RH3	R3		
0 1 0 0	RH4	R4	RR4	RQ4
0 1 0 1	RH5	R5		
0 1 1 0	RH6	R6	RR6	
0 1 1 1	RH7	R7		
1 0 0 0	RL0	R8	RR8	RQ8
1 0 0 1	RL1	R9		
1 0 1 0	RL2	R10	RR10	
1 0 1 1	RL3	R11		
1 1 0 0	RL4	R12	RR12	RQ12
1 1 0 1	RL5	R13		
1 1 1 0	RL6	R14	RR14	
1 1 1 1	RL7	R15		

Table 1

Register Organization (Continued)

Note that the byte register-addressing sequence (most significant bit distinguishes between the two bytes in a word register) is different from the memory addressing sequence (least significant bit distinguishes between the two bytes in a word). Long-word (32-bit) and quadruple-word (64-bit) registers are addressed by the binary number of their starting word registers (most significant word). For example, RR6 is addressed by a binary 6 and occupies word registers 6 and 7.

System/Normal Mode of Operation

The Z8000 CPU can run in one of two modes: System or Normal. In System Mode, all of the instructions can be executed and all of the CPU registers can be accessed. This mode is intended for use by programs that perform operating system type functions. In Normal Mode, some instructions, such as I/O instructions, are not all allowed, and the control registers of the CPU are inaccessible. In general, this mode of operation is intended for use by application programs. This separation of CPU resources promotes the integrity of the system since programs operating in Normal Mode cannot access those aspects of the CPU which deal with time-dependent or system interface events.

Normal Mode programs that have errors can always reproduce those errors for debugging purposes by simply re-executing the programs with their original data. Programs using facilities available only in System Mode may have errors due to timing considerations (e.g.,

Status Lines

The Z8000 CPU outputs status information over its four status lines (ST₀–ST₃) and the System/Normal line (S/ $\bar{N}$). This information can be used to extend the addressing range or to protect accesses to certain portions of memory. The types of status information and their codes are listed in Table 2.

Status conditions are mutually exclusive and can, therefore, be encoded without penalty. Most status definitions are self-explanatory. One code is reserved for future enhancements of the Z8000 Family.

Extension of the addressing range is accomplished in a Z8000 system by allocating physical memory to specific usage (program vs. data space, for example) and using external circuitry to monitor the status lines and select the appropriate memory space for each address. For example, the direct addressing range of the Z8002 CPU is limited to 64K bytes; however, a system can be configured

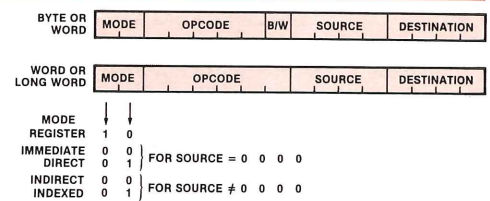


Figure 4. Instruction Format

based on the frequency of disk requests and disk arm position) that are harder to debug because these errors are not easily reproduced. Thus a preferred method of program development would be to partition the task into that portion which can be performed without recourse to resources accessible only in System Mode (which will usually be the bulk of the task) and that portion requiring System Mode resources. The classic example of this partitioning comes from current minicomputer and mainframe systems: the operating system runs in System Mode and the individual users write their programs to run in Normal Mode.

To further support the System/Normal Mode dichotomy, there are two copies of the stack pointer—one for the System Mode and another for Normal. Although the stacks are separated, it is possible to access the normal stack registers while in the System Mode by using the LDCTL instruction.

with 128K bytes if additional logic is used, say, to select the lower 64K bytes for program references and the upper 64K bytes for data references.

ST ₃ –ST ₀	Definition
0 0 0 0	Internal operation
0 0 0 1	Memory refresh
0 0 1 0	I/O reference
0 0 1 1	Special I/O reference
0 1 0 0	Segment trap acknowledge
0 1 0 1	Non-maskable interrupt acknowledge
0 1 1 0	Non-vectored interrupt acknowledge
0 1 1 1	Vectored interrupt acknowledge
1 0 0 0	Data memory request
1 0 0 1	Stack memory request
1 0 1 0	Data memory request (EPU)
1 0 1 1	Stack memory request (EPU)
1 1 0 0	Instruction space access
1 1 0 1	Instruction fetch, first word
1 1 1 0	Extension processor transfer
1 1 1 1	Reserved

Table 2

Status Lines
(Continued)

Protection of memory by access types is accomplished similarly. The memory is divided into blocks of locations and associated with each block is a set of legal status signals. For each access to the memory, the external circuit checks whether the CPU status is appropriate for the memory reference. The Z8010 Memory Management Unit is an example of an external memory-protection circuit, and it is discussed later in this tutorial.

The first word in an instruction fetch has its

own dedicated status code, namely 1101. This allows the synchronization of external circuits to the CPU. During all subsequent fetch cycles within the same instruction (remember, the longest instruction requires a total of four word fetches), the status is changed from 1101 to 1100. Load Relative and Store Relative also have a status of 1100 with the data reference, so information can be moved from program space to data space.

Refresh

The idea of incorporating the Refresh Counter in the CPU was pioneered by the Z-80 CPU, which performs a refresh access in a normally unused time slot after each opcode fetch. The Z8000 is more straightforward (each refresh has its own memory-access time slot of three clock cycles), and is more versatile (the refresh rate is programmable and capable of being disabled altogether).

The Refresh Register contains a 9-bit Row Counter, a 6-bit Rate Counter and an Enable Bit (Figure 5). The row section is output on AD₀-AD₈ during a refresh cycle. The Z8000 CPU uses word-organized memory, wherein A₀ is only employed to distinguish between the lower and upper bytes within a word during reading or writing bytes. A₀ therefore plays no role in refresh—it is always 0. The Row Counter is—at least conceptually—always incremented by two whenever the rate counter passes through zero. The Row Counter cycles through 256 addresses on lines AD₁-AD₈, which satisfies older and current 64- and 128-row addressing schemes, and can also be used with 256-row refresh schemes for 64K RAMs.

The Rate Counter determines the time between successive refreshes. It consists of a programmable 6-bit modulo-n prescaler

(n = 1 to 64), driven at one-fourth the CPU clock rate. The refresh period can be programmed from 1 to 64 μ s with a 4 MHz clock. A value of zero in the counter field indicates the maximum time between refreshes; a value of n indicates that refresh is to be performed every 4n clock cycles. Refresh can be disabled by programming the Refresh Enable Bit to be zero.

A memory refresh occurs as soon as possible after the indicated time has elapsed. Generally, this means after the T₃ clock cycle of an instruction if an instruction execution has commenced. When the CPU does not have control of the bus (during the bus-request/bus-acknowledge sequence, for example), it cannot issue refresh commands. Instead, it has internal circuitry to record "missed" refreshes; when the CPU regains control of the bus it immediately issues the "missed" refresh cycles. The Z8001 and Z8002 CPU can record up to two "missed" refresh cycles.

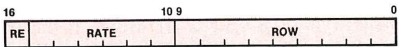


Figure 5. Refresh Counter

Instruction Prefetch (Pipelining)

Most instructions conclude with two or three clock cycles being devoted to internal CPU operations. For such instructions, the subsequent instruction-fetch machine cycle is overlapped with the concluding operations, thereby improving performance by two or three clock cycles per instruction.

Examples of instructions for which the subsequent instruction is fetched while they complete are Arithmetic and Shift instructions.

Some instructions for which the overlap is logically impossible are the Jump instructions (because the following instruction location has not been determined until the instruction completes). Some instructions for which overlap is physically impossible are the Memory Load instructions (because the memory is busy with the current instruction and cannot service the fetch of the succeeding instruction).

The Z8000 architecture has a mechanism for extending the basic instruction set through the use of external devices. Special opcodes have been set aside to implement this feature. When the CPU encounters instructions with these opcodes in its instruction stream, it will perform any indicated address calculation and data transfer, but otherwise treat the "extended instruction" as being executed by the external device. Fields have been set aside in these extended instructions which can be interpreted by external devices (called Extended Processing Units—EPU's) as opcodes. Thus by using appropriate EPU's, the instruction set of the Z8000 can be extended to include specialized instructions.

In general, an EPU is dedicated to performing complex and time consuming tasks in order to unburden the CPU. Typical tasks suitable for specialized EPU's include floating-point arithmetic, data base search and maintenance operations, network interfaces, graphics support operations—a complete list would include most areas of computing. EPU's are generally designed to perform their tasks on data resident in their internal registers. Moving information into and out of the EPU's internal registers, as well as instructing the EPU as to what operations are to be performed, is the responsibility of the CPU.

For the Z8000 CPU, control of the EPU's takes the following form. The Z8000 CPU fetches instructions, calculates the addresses of operands residing in memory, and controls the movement of data to and from memory. An EPU monitors this activity on the CPU's AD lines. If the instructions fetched by the CPU are extended instructions, all EPU's and the CPU latch the instruction (there may be several different EPU's controlled by one CPU). If the instruction is to be executed by a particular EPU, both the CPU and the indicated EPU will be involved in executing the instruction.

If the extended instruction indicates a transfer of data between the EPU's internal registers and the main memory, the CPU will calculate the memory address and generate the appropriate timing signals ($\overline{AS}$, $\overline{DS}$, $\overline{MREQ}$, etc.), but the data transfer itself is between the memory and the EPU (over the

AD lines). If a transfer of data between the CPU and EPU is indicated, the sender places the data on the AD lines and the receiver reads the AD lines during the next clock period.

If the extended instruction indicates an internal operation to be performed by the EPU, the EPU begins execution of that task and the CPU is free to continue on to the next instruction. Processing then proceeds simultaneously on both the CPU and the EPU until a second extended instruction is encountered that is destined for the same EPU (if more than one EPU is in the system, all can be operating simultaneously and independently). If an extended instruction specifies an EPU still executing a previous extended instruction, the EPU can suspend instruction fetching by the Z8000 CPU until it is ready to accept the next extended instruction: the mechanism for this is the $\overline{STOP}$ line, which suspends CPU activity during the instruction fetch cycle.

There are four types of extended instructions in the Z8000 CPU instruction repertoire: EPU internal operations; data transfers between memory and EPU; data transfers between EPU and CPU; and data transfer between EPU flag registers and CPU flag and control word. The last type is useful when the program must branch based on conditions determined by the EPU. Six opcodes are dedicated to extended instructions: 0E, 0F, 4E, 4F, 8E and 8F (in hexadecimal). The action taken by the CPU upon encountering these instructions is dependent upon an EPU control bit in the CPU's FCW. When this bit is set, it indicates that the system configuration includes EPU's; therefore, the instruction is executed. If this bit is clear, the CPU traps (extended instruction trap), so that a trap handler in software can emulate the desired operation.

In conclusion, the major features of this capability are, that multiple EPU's can be operating in parallel with the CPU, that the five main CPU addressing modes (Register, Immediate, Indirect Register, Direct Address, Indexed) are available in accessing data for the EPU; that each EPU can have more than 256 different instructions; and that data types manipulated by extended instructions can be up to 16 words long.

**Program
Status
Information**

The Program Status Information consists of the Flag And Control Word (FCW) and the Program Counter (PC). The Z8000 CPU uses one byte in FCW to store flags and another byte to store control bits.

Arithmetic Flags. Flags occupy the low byte in the FCW and are loaded, read, set and reset by the special instruction LDCTLB, RESFLG and SETFLG. The flags are:

- C** Carry
- Z** Zero
- S** Sign (1 = negative; two's complement notation is used for all arithmetic on data elements)
- P/V** Even Parity or Overflow (the same bit is shared)
- D** Decimal Adjust (differentiates between addition and subtraction)
- H** Half Carry (from the low-order nibble)

Control Bits. The control bits occupy the upper byte in the FCW. They are loaded and read by the LDCTL instruction, which is privileged in that it can be executed only in the System Mode. The control bits are:

- NVIE** Non-Vectored Interrupt Enable
- VIE** Vectored Interrupt Enable
- S/N** System or Normal Mode
- SEG** Segmented Mode Enable (Z8001 only)

The SEG bit is always 0 in the Z8002 even if the programmer attempts to set it. In the Z8001, a 1 in this bit indicates segmented operation. A 0 in the Z8001 SEG bit forces non-segmented operation and the CPU interprets all code as non-segmented. Thus, the Z8001 can execute modules of user code developed for the non-segmented Z8002.

**Interrupt
and Trap
Structure**

The Z8000 provides a powerful interrupt and trap structure. Interrupts are external asynchronous events requiring CPU attention, and are generally triggered by peripherals needing service. Traps are synchronous events resulting from the execution of certain instructions. Both are processed in a similar manner by the CPU.

The CPU supports three types of interrupts

(non-maskable, vectored and non-vectored), three internal traps (system call, unimplemented instruction, privileged instruction) and a segmentation trap. The vectored and non-vectored interrupts are maskable.

The descending order of priority for traps and interrupts is: internal traps, non-maskable interrupts, segmentation trap, vectored interrupts and non-vectored interrupts.

**Effects of
Interrupts
on Program
Status**

The Flag and Control Word and the Program Counter are collectively called the *Program Status Information*—a useful grouping because both the FCW and PC are affected by interrupts and traps. When an interrupt or trap occurs, the CPU automatically switches to the System Mode and saves the Program Status plus an identifier word on the system stack. The identifier supplies the reason for the interrupt. (The Z8002 pushes three words on the stack; the Z8001 pushes four words.)

After the pre-interrupt or "old" Program Status has been stored, the "new" Program Status is automatically loaded into the FCW and PC. This new Program Status Information is obtained from a specified location in memory, called the Program Status Area.

The Z8000 CPU allows the location of the Program Status Area anywhere in the addressable memory space, although it must be aligned to a 256-byte boundary. Because the Status Line code is 1100 (program reference) when the new Program Status is loaded, the Program Status must be located in program memory space if the memory uses this attribute (for example, when using the Z8010 Memory Management Unit or when separate memory modules are used for program and for data).

The Program Status Area Pointer (PSAP) specifies the beginning of the Program Status Area. In the Z8002, the PSAP is stored in one word, the lower byte of which is zero. The Z8001, however, stores its PSAP in two words. The first contains the segment number and the second contains the offset, the lower byte of which is again zero. The PSAP is loaded and read by the LDCTL instruction.

In the Z8002, the first 14 words (28 bytes) of the Program Status Area contain the Program Status Information for the following interrupt conditions:

Location (In Bytes)	Condition
0-3	Not used (reserved for future use)
4-7	Unimplemented instruction has been fetched, causing a trap
8-11	Privileged instruction has been fetched in Normal Mode, causing a trap
12-15	System Call instruction
16-19	Not used
20-23	Non-maskable interrupt
24-27	Non-vectored interrupt

Effects of Interrupts on Program Status

(Continued)

Bytes 28-29 contain the FCW that is common to all vectored interrupts. Subsequent locations contain the vector jump table (new PC for vectored interrupts). These locations are addressed in the following way: the 8-bit vector that the interrupting device has put on the lower byte of the Address/Data bus (AD_0-AD_7) is doubled and added to $PSAP + 30$. Thus,

Vector 0 addresses $PSAP + 30$,
 Vector 1 addresses $PSAP + 32$, and
 Vector 255 addresses $PSAP + 540$.

In the segmented Z8001, the first 28 words of the Program Status Area (56 bytes) contain the Program Status Information (reserved word, FCW, segment number, offset), for the following interrupt conditions:

Location (In bytes)	Condition
0-7	Not used (reserved for future use)
8-15	Unimplemented instruction has been fetched causing a trap
16-23	Privileged instruction has been fetched in Normal Mode causing a trap
24-31	System Call instruction
32-39	Segmentation trap (memory violation detected by the Z8010 Memory Management Unit)
40-47	Non-maskable interrupt
48-55	Non-vectored interrupt

Bytes 56-59 contain the reserved word and FCW common to all vectored interrupts. Subsequent locations contain the vector jump table (the new segment number and offset for all vectored interrupts). These locations are addressed in the following way: the 8-bit vector that the interrupting device has put on the lower byte of the Address/Data bus (AD_0-AD_7) is doubled and added to $PSAP + 60$. Thus,

Vector 0 addresses $PSAP + 60$,
 Vector 2 addresses $PSAP + 64$, and
 Vector 254 addresses $PSAP + 568$.

Care must be exercised in allocating vector locations to interrupting devices; always use even vectors. Thus there are effectively only 128 entries in the vector jump table. (Figure 6 illustrates the Program Status Area.)

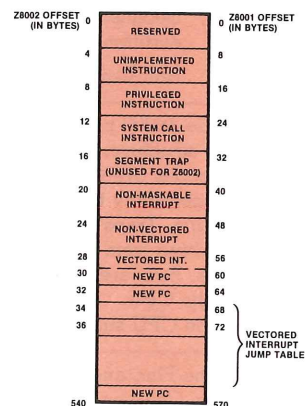


Figure 6. Program Status Area

The way a processor addresses and manages its memory is an important aspect in both the evaluation of the processor and the design of a computer system that uses the processor. Z8000 architecture provides a consistent memory address notation in combining bytes into words and words into long words. All three data types are supported for operands in the Z8000 instruction set. I/O data can be either byte- or word-oriented.

The Z8001 CPU provides a segmented addressing space with 23-bit addressing. The Z8010 Memory Management Unit can increase the address range of this processor. To support a memory management system, the Z8001 processor generates Processor Status Information.

These signals are also generated by the Z8002 CPU and—as mentioned earlier—can be used to increase the address range of this processor beyond its nominal 64K byte limit. It is not necessary to use a Z8010 Memory Management Unit with a Z8001. The segment number (upper six bits of the address) can be used directly by the memory system as part of the absolute address.

These issues are discussed in more detail in the following sections, along with a description of the method used to encode certain segmented addresses into one word. A brief comment on the use of 16K Dynamic RAMs with the Z8001 concludes this group of sections that deal with Z8000 CPU memory features.

ing and sorting of byte and word strings.

Bit labeling within a byte does not follow this order. The least significant bit in a byte, word or long word is called Bit 0 and occurs in the byte with the highest memory address. This is consistent with the convention where bit n corresponds to position 2ⁿ in the conventional binary notation. This ordering of bit numbers is also followed in the registers.

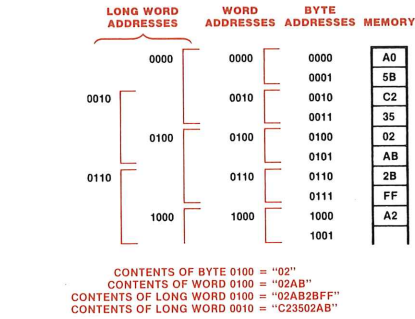


Figure 7. Memory Addressing

Memory and I/O Addressing

Like most 16-bit microprocessors, the Z8000 CPU uses a 16-bit parallel data bus between the CPU and memory or I/O. The CPU is capable of reading or writing a 16-bit word with every access. Words are always addressed with even addresses ($A_0 = 0$). All instructions are words or multiple words.

The Z8000 CPU can, however, also read and write 8-bit bytes, so memory and I/O addresses are always expressed in bytes. The Byte/Word ($B/\bar{W}$) output indicates whether a byte or word is addressed (High = byte). A_0 distinguishes between the upper and lower byte in memory or I/O. The most significant byte of the word is addressed when A_0 is Low (Figure 8).

For word operations in both the read and write modes, $B/\bar{W} = \text{Low}$, A_0 is simply

ignored and A_1 – A_{15} address the memory or I/O. For byte operations in the read mode, $B/\bar{W} = \text{High}$, A_0 is again ignored, and a whole word (both bytes) is read, but the CPU internally selects the appropriate byte. For byte operations in the write mode, the CPU outputs identical information on both the Low (AD_0 – AD_7) and the High (AD_8 – AD_{15}) bytes of the Address/Data bus. External TTL logic must be used to enable writing in one memory byte and disable writing in the other byte, as defined by A_0 . The replication of byte information for writes is for the current implementation and may change for subsequent Z8000 CPUs; therefore system designs should not depend upon this feature.

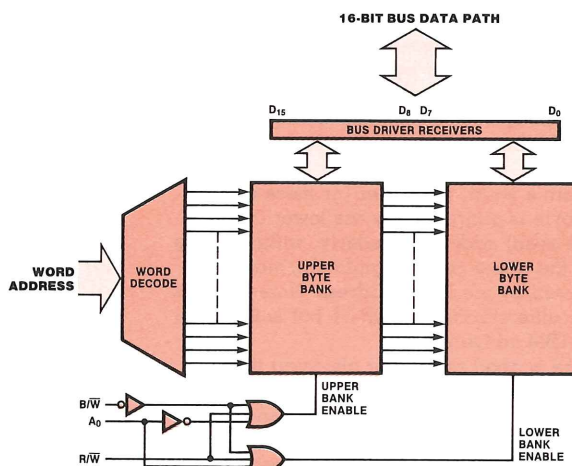


Figure 8. Byte/Word Selection

Segmentation In organizing memory, segmentation is a powerful and useful technique because it forms a natural way of dividing an address space into different functional areas. A program typically partitions its available memory into disjointed areas for particular uses. Examples of this are storing the procedure instructions, holding its global variables, or serving as a buffer area for processing large, disk-resident data bases. The requirements for these different areas may differ, and the areas themselves may be needed only part of the time.

Segmentation reflects this use of memory by allowing a user to employ a different segment for each different area. A memory management system can then be employed to provide system support, such as swapping segments from disk to primary memory as requested (as in overlays), or in monitoring memory accesses and allowing only certain types of accesses to

a particular segment. Thus, dealing with segments is a convenient way of specifying portions of a large address space.

When segmentation is combined with an address translation mechanism to provide relocation capability, the advantages of segmentation are enhanced. Now segments can be of variable user-specifiable sizes and located anywhere in memory.

The Z8001 generates 23-bit logical addresses, consisting of a 7-bit segment number and a 16-bit offset. Thus each of its six memory address spaces consists of 128 segments, and each segment can be up to 64K bytes. Different routines of a program can reside in different segments, and different data sets can reside in different segments. The Z8010 Memory Management Unit translates these logical addresses into physical-memory locations.

Long Offset and Short Offset Addressing

When a segmented address is stored in memory or in a register, it occupies two 16-bit words as previously described for the PC and PSAP. This is a consequence of the large addressing range. When a segmented address is part of an instruction in the Direct Address and Indexed Address Modes, there are two representations: Long and Short Offset addressing.

In the general unrestricted case of Long Offset, the segmented address occupies two words, as described before. The most significant bit in the segment word is a 1 in this case.

The Short Offset Mode squeezes the segment number and offset into one word, saving pro-

gram size and execution time. Since 23 bits obviously don't fit into a 16-bit word, the 8 most significant bits of the offset are omitted and implied to be zero. The most significant bit of the address word is made 0 to indicate Short Offset Mode. Short Offset addresses are thus limited to the first 256 bytes at the beginning of each segment. This may appear to be a severe restriction, but it is very useful, especially in the Index Mode, where the index register can always supply the full 16-bit range of the offset. Short Offset saves one instruction word and speeds up execution by two clock cycles in Direct Address Mode and three clock cycles in Indexed Mode.

Using the Z8010 Memory Management Unit

The Z8001 CPU can be combined with another 48-pin LSI device—the Z8010 MMU—for sophisticated memory management. The MMU provides address translation from the logical addresses generated by the Z8001 CPU to the physical addresses used by the memory. An address translation table, containing starting addresses and size information for each of the 64 segments, is stored in the MMU. The translation table can be written and read by the CPU using Special I/O instructions. The MMU thus provides address relocation under software control, making software addresses (i.e., logical addresses) independent of the physical memory addresses.

But the MMU provides much more than address relocation; it also monitors and protects memory access. The MMU provides a Trap input to the CPU and—if necessary—an inhibit signal (SUP) to the memory write logic when specific memory-access violations occur. The MMU provides the following types of memory protection:

- Accesses outside the segment's allotted memory can be prevented.
- Any segment can be declared invalid or non-accessable to the CPU.
- Segments can be declared Read Only.
- By designating a segment as System Only, access can be prohibited during the Normal Mode.
- Declaring a segment Execute Only means it can be accessed only during instruction access cycles. Data or stack use is prohibited.
- Any segment can be excluded from DMA access.
- Segments can have a Direction And Write Warning attribute, which generates a trap when a write access is made in the last 256 bytes of its size. This mechanism can be used to prevent stack overflow.

Multiple MMUs must be used when more than 64 segments are needed. Thus, to support the full complement of 128 segment numbers provided for each Z8001 CPU address space, two MMUs are required. The MMU has been designed for multiple-chip configurations, both to support 128-segment translation tables and to support multiple translation table systems.

Note that the memory management features do not interfere with the ability to directly address the entire memory space. Once programmed, the MMU (or MMUs) translates and monitors any memory address generated by the CPU.

The MMU contains status bits that describe the history of each segment. One bit for each segment indicates whether the segment has been accessed; another bit indicates whether the segment has been written. This is important for certain memory management schemes. For example, the MMU indicates which segments have been updated and, therefore, must be saved on disk before the memory can be used by another program.

When translating logical addresses to physical memory addresses, the MMU must do the following: access its internal 64×32 -bit RAM, using the segment number as the address, then add the 16 bits of RAM output to the most significant address byte (AD₈-AD₁₅) and finally place the result on its Address outputs. The least significant byte (AD₀-AD₇) bypasses the MMU.

The internal RAM access time is approximately 150 ns. Throughput delay is avoided by making the segment number available early: SN₀-SN₇ are output one clock period earlier than the address information on AD₀-AD₇.

In summary, the Z8000 CPU supports sophisticated memory management through such architectural features as the Status Lines, the R/W and S/N lines, Segment Trap input line, and early output of segment numbers.

**Using 16K
Dynamic
RAMs with
the Z8001**

Z8000 systems usually implement most of their memory with $16K \times 1$ -bit dynamic RAMs that have time-multiplexed addresses (Zilog also manufactures this device—the Z6116). In Z8001-based systems with MMUs, CPU Address/Data lines AD₁–AD₇ supply row addresses, MMU address outputs A₈–A₁₄ supply column addresses, and MMU outputs A₁₅–A₂₃ are decoded to generate Chip Select signals that gate either RAS or CAS or both.

Gating RAS reduces power consumption because all non-selected memories remain in the standby mode. But this technique

requires that $\overline{\text{RAS}}$ must wait for the availability of the most significant address bits from the MMU. During refresh, the $\overline{\text{RAS}}$ decoder must be changed to activate all memories simultaneously.

Gating $\overline{\text{CAS}}$ does not achieve lower power consumption; however, this technique allows the use of slower memories because RAS can be activated as soon as the CPU address outputs are stable, without waiting for the MMU delay. Also, there is no need to change the $\overline{\text{CAS}}$ decoder during refresh.

**Data Types
and
Instructions**

The Z8000 architecture directly supports bits, digits, bytes, and 16- or 32-bit integers as primitive operands in its instruction set. In addition, the rich set of addressing modes supports higher-level data constructs such as arrays, lists and records. The Z8000 also intro-

duces a number of powerful instructions that extend the capabilities of microprocessors. The remaining sections of this paper describe Z8000 data types, addressing modes, and a selection of novel instructions.

Data Types

Operands are 1, 4, 8, 16, 32, or 64 bits, as specified by the instruction. In addition, strings of 8- or 16-bit data can be manipulated by single instructions. Of particular interest are the increased precisions of the arithmetic instructions. Add and Subtract instructions can

operate on 8-, 16-, or 32-bit operands; Multiply instructions can operate on 16- or 32-bit multiplicands; and Divide instructions can operate on 32- or 64-bit dividends. The Shift instructions can operate on 8-, 16-, and 32-bit registers.

**Addressing
Modes**

The rich variety of addressing modes offered by Z8000 architecture includes: Register, Immediate, Indirect Register, Direct Address, Index, Relative Address, Base Address, and Base Index. Three are of particular interest with respect to high-level data structures: Indirect Register, Base Address, and Base Index. These modes can be used for lists, records, and arrays, respectively.

Indirect Register. In this addressing mode, the contents of the register are used as a memory address. This mode is needed whenever special address arithmetic must be performed to reference data. Essentially, the address is calculated in a register and then used to fetch the data. For example, this mode is useful when manipulating a linked list, where each entry contains a memory pointer to the memory location of the next entry. Essentially, the pointer is loaded into a register and used to access the next item on the list. When the list item is large or has a complex structure, the Base Address or Base Index Modes can be used to access various components of the item.

Base Address. In this addressing mode, the memory address contained in the register (the base) is modified by a displacement in the instruction (known at compile time). This mode

is useful, for example, in accessing fields within a record whose format is fixed at compile time.

Base Index. The memory address in this addressing mode is contained in a register (the base) and is modified by the contents of another register (the index). This mode can be useful in accessing the components of an array, because the index of the component is usually calculated during execution time—as a function of the index of a DO-Loop, for example.

Index vs. Base Address. In the Z8002 and in the Z8001 running non-segmented, these two addressing modes are functionally equivalent, because the base address and displacement are both 16-bit values.

When the Z8001 runs segmented, there is a difference: in the Index mode, the base address (including the segment number) is contained in the instruction, in either Short Offset or Long Offset notation. The 16-bit displacement stored in a register is then added to the offset in the base address to calculate the effective address. In the Base Address Mode, on the other hand, the 16-bit displacement is specified in the instruction and is added to the offset of the base address that is stored in a long-word register.

The Z8000 offers an abundant instruction set that represents a major advance over its predecessors. The Load and Exchange instructions have been expanded to support operating system functions and conversion of existing microprocessor programs. The usual Arithmetic instructions can now deal with higher-precision operands, and hardware Multiply and Divide instructions have been added. The Bit Manipulation instructions can access a calculated bit position within a byte or word, as well as specify the position statically in the instruction.

The Rotate and Shift instructions are considerably more flexible than those in previous microprocessors. The String instructions are useful in translating between different character codes. Special I/O instructions are included to manage peripheral devices, such as the Memory Management Unit, that do not respond to regular I/O commands. Multiple-processor configurations are supported by special instructions.

The following instructions exemplify the innovative nature of the Z8000 instruction set. A complete list of Z8000 instructions can be found in the reference materials listed at the end of this tutorial.

Load and Exchange Instructions.

Exchange Byte (EX) is practical for converting Z-80, 8080, 6800 and other microprocessor programs into Z8000 code, because the Z8000 uses the opposite assignment of odd/even addresses in 16-bit words.

Load Multiple (LDM) saves *n* registers and is useful for switching tasks.

Load Relative (LDR) loads fixed values from program space into data space.

Arithmetic Instructions.

Add With Carry and Subtract With Carry (ADC, SBC) are conventionally used in 8-bit microprocessors for multiprecision arithmetic operations. These instructions are rarely used with the Z8000 CPU because it has 16- and 32-bit arithmetic instructions.

Decrement By N and Increment By N (DEC, INC) are intended for address and pointer manipulation, but can also be used for Quick Add/Subtract Immediate with 4-bit nibbles. The flag setting is different from Add/Subtract instructions—as is conventional—in that the Carry and Decimal adjust flags are unaffected by the Increment and Decrement instructions to support multiple precision arithmetic.

Decimal Adjust (DAB) automatically generates the proper 2-digit BCD result after a byte Add or Subtract operation, and eliminates the need for special decimal arithmetic instructions.

Multiply (MULT) provides signed (two's complement) multiplication of two words, generating a long-word result; or of two long-words generating a quadruple word result. No byte multiply exists because it is rarely used and, after sign extension, can be performed by a word multiply.

Divide (DIV) provides signed (two's complement) division of a long word by another word, generating a word quotient and a remainder word; or of one quadruple-word by a long-word, generating a long-word quotient and long-word remainder.

Both Multiply and Divide use a conforming register assignment. That is, a multiply followed by a divide on the same registers is essentially a no-op. The register designation used in the operation description must be even for word operations and must be a multiple of four for long-word operations.

Logical Instructions.

Test Condition Code (TCC) performs the same test as a Jump instruction, but affects the least significant bit of a specified register instead of changing the PC.

Program Control Instructions.

Call Relative (CALR) is a shorter, faster version of Call, but with a limited range.

Decrement And Jump If Non-Zero (DJNZ) is a one-word basic looping instruction.

Jump Relative (JR) is a shorter, faster version of Jump, but with a limited range.

Bit Manipulation Instructions.

Test Bit, Reset Bit, Set Bit (BIT, RES, SET) are available in two forms: static and dynamic. For the static form, any bit (the position is defined in the immediate word of the instruction) located in any byte or word in any register or in memory can be set, reset or tested (inverted and routed into the Z flag).

For the dynamic form, any bit (the position is defined by the content of a register that is, in turn, specified in the instruction) located in any byte or word in any register, but not in memory, can be set, reset or tested.

Test And Set (TSET) is a read/modify/write instruction normally used to create operating system locks. The most significant bit of a byte or word in a register or in memory is routed into the S flag bit and the whole byte or word is then set to all 1s. During this instruction, the processor does not relinquish the bus.

Test Multi-Micro Bit and Multi-Micro Request/Set/Reset (MBIT, MREQ, MSET, MRES) are used to synchronize the access by multiple microprocessors to a shared resource,

The Instruction Set
(Continued)

such as a common memory, bus, or I/O device.

Note that the instruction MREQ (Multi-Microprocessor Request) has nothing whatsoever in common with the MREQ (Memory request) output from the Z8000 CPU.

Rotate and Shift Instructions.

The Z8000 CPU has a complete set of shift instructions that shift any combination of bytes or words, right or left, arithmetically or logically, by any meaningful number of positions as specified either in the instruction (static) or in a register (dynamic).

The CPU also has a smaller repertoire of rotate instructions that rotates bytes or words, either right or left, through carry or not, and by one bit or by two bits.

The instructions Rotate Digit Left and Rotate Digit Right (RLDB, RRDB) rotate 4-bit BCD digits right or left, and are used in BCD arithmetic operations.

Block Transfer and String Manipulation Instructions.

Translate And Decrement/Increment (TRDB, TRIB) is used for code conversion, such as ASCII to EBCDIC. These instructions translate a byte string in memory by substituting one string by its table-lookup equivalent. TRDB and TRIB execute one operation and decrement the contents of the length register; thus they are useful as part of loop performing several actions on each character.

Translate, Decrement/Increment and Repeat (TRDRB, TRIRB) are the same as TRDB and

TRIB, except they repeat automatically until the contents of the length register become zero. They are therefore useful in straightforward translation applications.

Translate And Test, Decrement/Increment (TRTDB, TRTIB) tests a character according to the contents of the translation table.

Translate And Test, Decrement/Increment And Repeat (TRTDRB, TRTIRB) scans a string of characters. The first character is tested and, depending on the contents of the translation table, the process stops or skips to the next character. Stopped characters can be used for further processing.

I/O and Special I/O Instructions.

The Z8000 CPU has two complete sets of I/O instructions: Standard I/O and Special I/O. The only difference is the status information on the ST₀-ST₃ outputs. Standard I/O instructions are used to communicate with Z-Bus compatible peripherals. Special I/O instructions are typically used for communicating with the Memory Management Unit.

Both types of instructions transfer 8 or 16 bits and use a type of 16-bit addressing analogous to the Z8002 memory-addressing scheme: For word operations, A₀ is always zero; in byte-input operations, A₀ is used internally by the CPU to select the appropriate byte; in byte-output operations, the byte is duplicated in the high and low bytes of the address/data bus, and external logic uses A₀ to enable the appropriate output device.

Bibliography

Selected Publications on the Z8000 Family
Z8001/Z8002 CPU Product Specification
(03-8002-01)
Z8000 CPU Instruction Set (03-8020-01)

*Z8000 PLZ/ASM Assembly Language
Programming Manual* (03-3055-01)
Z8010 MMU Product Specification (03-8070-01)
Z6116 RAM Product Specification (03-8045-01)

The Zilog Z-Bus



NEW
1980

Summary

Preliminary

February 1980

Features

- Multiplexed address/data bus shared by memory and I/O transfers.
- 16 or more memory address bits; 16-bit I/O addresses; 8 or 16 data bits.
- Supports polling and vectored or non-vectored interrupts.
- Daisy-chain interrupt structure services interrupts without a separate priority controller.

- Direct addressing of registers within a peripheral facilitates I/O programming.
- Bus signals allow asynchronous CPU and peripheral clocks.
- Daisy-chain bus-request structure supports distributed control of the bus.
- Shared resources can be managed by a general-purpose, distributed resource-request mechanism.

General Description

The Z-Bus is a high-speed parallel shared bus that links components of the Z8000 Family. It provides family members with a common communication interface that supports the following kinds of interactions:

- *Data Transfer.* Data can be moved between bus controllers (such as a CPU) and memories or peripherals.
- *Interrupts.* Interrupts can be generated by peripherals and serviced by CPUs over the bus.
- *Resource Control.* Distributed management of shared resources (including the bus itself) is supported by a daisy-chain priority mechanism.

The heart of the Z-Bus is a set of multiplexed address/data lines and the signals that control these lines. Multiplexing data and addresses onto the same lines makes more efficient use of pins and facilitates expansion of the number of data and address bits. Multiplexing also allows straightforward addressing of a peripheral's internal registers, which greatly simplifies I/O programming.

A daisy-chained priority mechanism resolves interrupt and resource requests, thus allowing distributed control of the bus and eliminating the need for separate priority controllers. The resource-control daisy chain allows wide physical separation of components.

The Z-Bus is asynchronous in the sense that peripherals do not need to be synchronized with the CPU clock. All timing information is provided by Z-Bus signals.

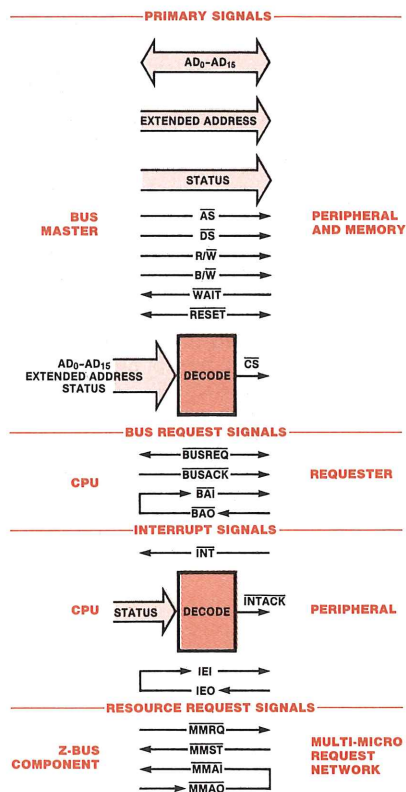


Figure 1. Z-Bus Signals

Z-Bus Components

A Z-Bus component is one that uses Z-Bus signals and protocols, and meets the specified ac and dc characteristics. Most components in the Z8000 Family are Z-Bus components. The four categories of Z-Bus components are as follows:

CPUs. A Z-Bus system contains one CPU, and this CPU has default control of the bus and typically initiates most bus transactions. Besides generating bus transactions, it handles interrupt and bus-control requests. The Z8001 Segmented CPU and Z8002 Non-Segmented CPU are Z-Bus CPUs.

Peripherals. A Z-Bus peripheral is a component capable of responding to I/O transactions and generating interrupt requests. The Z8036 Counter Input/Output Circuit (Z-CIO),

Z8038 FIFO Input/Output, Interface Unit (Z-FIO), the Z8030 Serial Communication Controller (Z-SCC), the Z8034 Universal Peripheral Controller (Z-UPC), and the Z8052 CRT Controller (Z-CRT) are all Z-Bus peripherals.

Requesters. A Z-Bus requester is any component capable of requesting control of the bus and initiating transactions on the bus. A Z-Bus requester is usually also a peripheral. The Z8016 DMA Transfer Controller (Z-DTC) is a Z-Bus requester and a peripheral.

Memories. A Z-Bus memory is one that interfaces directly to the Z-Bus and is capable of fetching and storing data in response to Z-Bus memory transactions. The Z6132 Quasi-Static RAM is a Z-Bus memory.

Other Components

The Z8 Microcomputer—in its micro-processor configuration—conforms to Z-Bus timing (which allows it to use Z-Bus peripherals and memories), but is missing a wait input and certain status outputs.

The Z8010 Memory Management Unit (Z-MMU) is a Z8000 CPU support component that interfaces with part of the Z-Bus on the CPU side and provides demultiplexed

addresses on the memory side.

The Z8060 First-In-First-Out Buffer (Z-FIFO) is not a Z-Bus component; rather, it is used to expand the buffer depth of the Z-FIO or to interface the I/O ports of the Z-UPC, Z-CIO, or Z-FIO to user equipment.

Z-80 Family components, while not Z-Bus compatible, are easily interfaced to Z-Bus CPUs.

Operation

Two kinds of operations can occur on the Z-Bus: transactions and requests. At any given time, one device (either the CPU or a bus requester) has control of the Z-Bus and is known as the *bus master*. A transaction is initiated by a bus master and is responded to by some other device on the bus. Four kinds of transactions occur in Z-Bus systems:

- **Memory.** Transfers 8 or 16 bits of data to or from a memory location.
- **I/O.** Transfers 8 or 16 bits of data to or from a peripheral.
- **Interrupt Acknowledge.** Acknowledges an interrupt and transfers an identification/status vector from the interrupting peripheral.
- **Null.** Does not transfer data. Typically used for refreshing memory.

Only one transaction can proceed on the

bus at a time, and it must be initiated by the bus master. A request, however, may be initiated by a component that does not have control of the bus. There are three kinds of requests:

- **Interrupt.** Requests the attention of the Z-Bus CPU.
- **Bus.** Requests control of the Z-Bus to initiate transactions.
- **Resource.** Requests control of a particular resource.

When a request is made, it is answered according to its type: for interrupt requests an interrupt-acknowledge transaction is initiated; for bus and resource requests an acknowledge signal is sent. In all cases a daisy-chain priority mechanism provides arbitration between simultaneous requests.

The Z-Bus consists of a set of common signal lines that interconnect bus components (Figure 1). The signals on these lines can be grouped into four categories, depending on how they are used in transactions and requests.

Primary Signals. These signals provide timing, control, and data transfer for Z-Bus transactions.

AD_0 – AD_{15} . Address/Data (active High). These multiplexed data and address lines carry I/O addresses, memory addresses, and data during Z-Bus transactions. A Z-Bus may have 8 or 16 bits of data depending on the type of CPU. In the case of an 8-bit Z-Bus, data is transferred on AD_0 – AD_7 .

Extended Address. (active High). These lines extend AD_0 – AD_{15} to support memory addresses greater than 16 bits. The number of lines and the type of address information carried is dependent on the CPU.

Status. (active High). These lines designate the kind of transaction occurring on the bus and certain additional information about the transaction (such as program or data memory access or System versus Normal Mode).

$\overline{AS}$. Address Strobe (active Low). The rising edge of $\overline{AS}$ indicates the beginning of a transaction and that the Address, Status, $R/\overline{W}$, and $B/\overline{W}$ signals are valid.

$\overline{DS}$. Data Strobe (active Low). $\overline{DS}$ provides timing for data movement to or from the bus master.

$R/\overline{W}$. Read/Write (Low = write). This signal determines the direction of data transfer for memory or I/O transactions.

$B/\overline{W}$. Byte/Word (Low = word). This signal indicates whether a byte or word of data is to be transmitted on a 16-bit bus. This signal is not present on an 8-bit bus.

$\overline{WAIT}$. (active Low). A Low on this line indicates that the responding device needs more time to complete a transaction.

$\overline{RESET}$. (active Low). A Low on this line resets the CPU and bus users. Peripherals may be reset by $\overline{RESET}$ or by holding $\overline{AS}$ and $\overline{DS}$ Low simultaneously.

$\overline{CS}$. Chip Select (active Low). Each peripheral or memory component has a $\overline{CS}$ line that is decoded from the address and status lines. A Low on this line indicates that the peripheral or memory component is being addressed by a transaction. The Chip Select information is latched on the rising edge of $\overline{AS}$.

Bus Request Signals. These signals make bus requests and establish which component should obtain control of the bus.

$\overline{BUSREQ}$. Bus Request (active Low). This line is driven by all bus requesters. A Low indicates that a bus requester has or is trying to obtain control of the bus.

$\overline{BUSACK}$. Bus Acknowledge (active Low). A Low on this line indicates that the Z-Bus CPU has relinquished control of the bus in response to a bus request.

$\overline{BAI}$, $\overline{BAO}$. Bus Acknowledge In, Bus Acknowledge Out (active Low). These signals form the bus-request daisy chain.

Z-Bus Connections	Signal	CPU	Requester	Peripheral	Memory
	$\overline{AD_0-AD_{15}}$	Bidirectional ² 3-state	Bidirectional ² 3-state	Bidirectional ¹ 3-state	Bidirectional ² 3-state
	Extended Address ⁸	Output 3-state	Output 3-state	□	Input
	Status	Output 3-state	Output 3-state	Input ¹⁰	□
	$R/\overline{W}$	Output 3-state	Output 3-state	Input	Input
	$B/\overline{W}$ ⁹	Output	Output	Input ³	Input
	$\overline{WAIT}$	Input	Input	Output ⁸ Open Drain	Output ⁸ Open Drain
	$\overline{AS}$	Output 3-state	Output 3-state	Input	Input
	$\overline{DS}$	Output 3-state	Output 3-state	Input	Input
	$\overline{CS^4}$	□	□	Input	Input
	$\overline{RESET}$	Input	Input	Input ⁵	□
	$\overline{BUSREQ}$	Input	Bidirectional Open Drain	□	□
	$\overline{BUSACK}$	Output	□	□	□
	$\overline{BAI^7}$	□	Input	□	□
	$\overline{BAO^7}$	□	Output	□	□
	$\overline{INT}$	Input	□	Output Open Drain	□
	$\overline{INTACK^6}$	□	□	Input ¹¹	□
	$\overline{IEI^7}$	□	□	Input	□
	$\overline{IEO^7}$	□	□	Output	□
	$\overline{MMRQ^{12}}$	Output Open Drain			
	$\overline{MMST^{12}}$	Input			
	$\overline{MMAI^7, 12}$	Input			
	$\overline{MMAO^7, 12}$	Output			

1. Only $\overline{AD_0-AD_7}$, unless peripheral is 16-Bit.

2. For an 8-bit bus, only $\overline{AD_0-AD_7}$ are bidirectional.

3. Only for a 16-bit peripheral.

4. Derived signal, one for each peripheral or memory; decoded from status and address lines.

5. Optional—peripherals are typically reset by $\overline{AS}$ and $\overline{DS}$ being Low simultaneously; however, they can have a reset input.

6. Derived signal; decoded from status lines.

7. Daisy-chain lines.

8. Optional signal(s).

9. For 16-bit data bus only.

10. Optional—usually only input on peripherals that are also requesters.

11. May be omitted if peripheral inputs status lines.

12. Optional signal; any component may attach to the resource request lines.

Table 1. Z-Bus Component Connections to Signal Lines. This table shows how the various Z-Bus components attach to each signal line. When a device is both a bus

requester and a peripheral, the attributes in both columns of the table should be combined (e.g., input combined with output and 3-state becomes bidirectional and 3-state.)

Signal Lines (Continued)

Interrupt Signals. These signals are used for interrupt requests and for determining which interrupting component is to respond to an acknowledge. To support more than one type of interrupt, the lines carrying these signals can be replicated. (The Z8000 CPU supports three types of interrupts: non-maskable, vectored, and non-vectored.)

$\overline{INT}$. *Interrupt (active Low).* This signal can be driven by any peripheral capable of generating an interrupt. A Low on $\overline{INT}$ indicates that an interrupt request is being made.

$\overline{INTACK}$. *Interrupt Acknowledge (active Low).* This signal is decoded from the status lines. A Low indicates an interrupt acknowledge transaction is in progress. This signal is latched by the peripheral on the rising edge of $\overline{AS}$.

$\overline{IEI}$, $\overline{IEO}$. *Interrupt Enable In, Interrupt Enable Out (active High).* These signals form the interrupt daisy chain.

Resource Request Signals. These signals are used for resource requests. To manage more than one resource, the lines carrying these signals can be replicated. (The Z8000 supports one set of resource request lines.)

$\overline{MMRQ}$. *Multi-Micro Request (active Low).* This line is driven by any device that can use the shared resource. A Low indicates that a request for the resource has been made or granted.

$\overline{MMST}$. *Multi-Micro Status (active Low).* This pin allows a device to observe the value of the $\overline{MMRQ}$ line. An input pin other than $\overline{MMRQ}$ facilitates the use of line drivers for $\overline{MMRQ}$.

$\overline{MMAI}$, $\overline{MMAO}$. *Multi-Micro Acknowledge In, Multi-Micro Acknowledge Out (active Low).* These lines form the resource-request daisy chain.

2

Transactions

All transactions start with Address Strobe being driven Low and then raised High by the bus master (Figure 2). The Status lines are valid on the rising edge of Address Strobe and indicate the type of transactions being initiated. If the transaction requires an address, it must also be valid on the rising edge of Address Strobe.

For all transactions except null transactions (which do nothing beyond this point), data is then transferred to or from the bus master. The

bus master uses Data Strobe to time the movement of data. For a read ($R/\overline{W} = \text{High}$), the bus master makes AD_0-AD_{15} inactive before driving Data Strobe Low so that the addressed memory or peripheral can put its data on the bus. The bus master samples this data just before raising Data Strobe High. For a write ($R/\overline{W} = \text{Low}$), the bus master puts the data to be written on AD_0-AD_{15} before forcing Data Strobe Low.

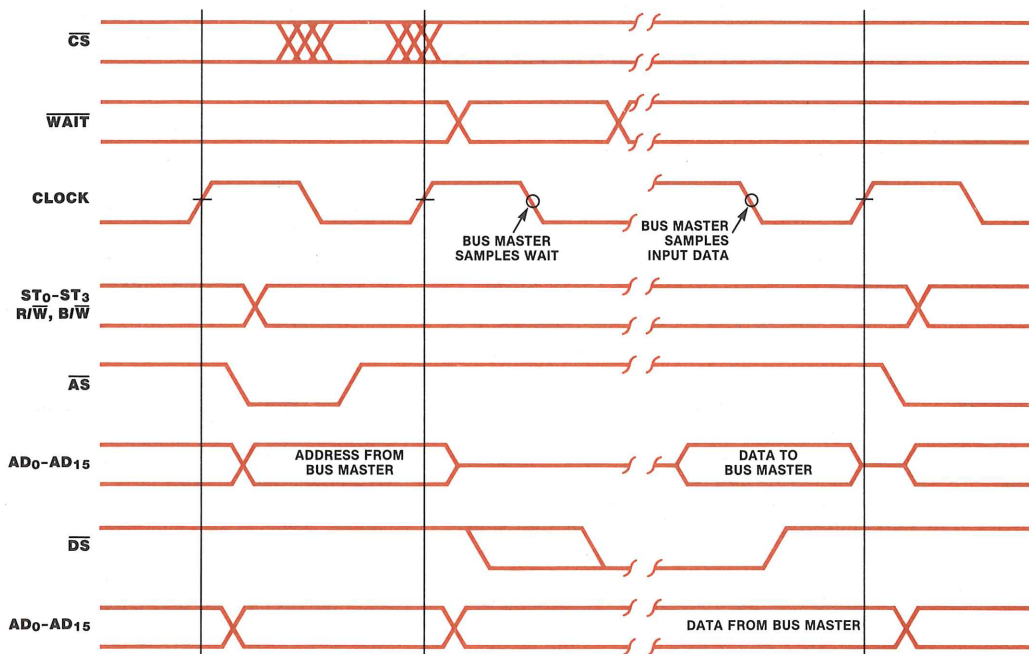


Figure 2. Typical Transaction Timing

Memory Transactions

For a memory transaction, the Status lines distinguish among various address spaces, such as program and data or system and normal, as well as indicating the type of transaction. The memory address is put on AD₀-AD₁₅ and on the extended address lines.

For a Z-Bus with 16-bit data, the memory is organized as two banks of eight bits each (Figure 3). One bank contains all the upper

bytes of all the addressable 16-bit words. The other bank contains all the lower bytes. When a single byte is written ($R/\overline{W}$ = Low, $B/\overline{W}$ = High), only the bank indicated by address bit A₀ is enabled for writing.

For a Z-Bus with 8-bit data, the memory is organized as one bank which contains all bytes. This bank always inputs and outputs its data on AD₀-AD₇.

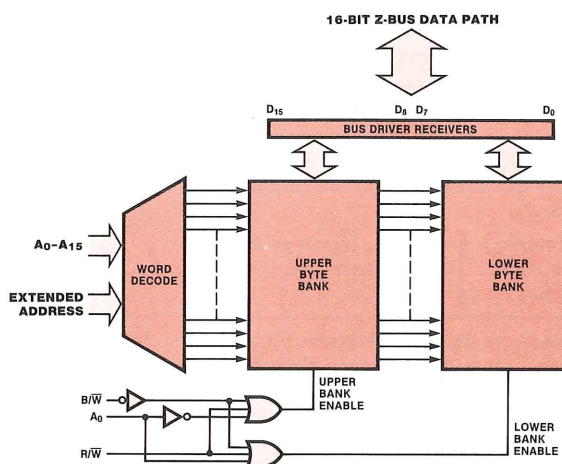


Figure 3. Byte/Word Memory Organization

I/O Transactions

I/O transactions are similar to memory transactions with two important differences. The first is that I/O transactions take an extra clock cycle to allow for slow peripheral operation. The second is that byte data (indicated by $B/\overline{W}$ High on a 16-bit bus) is always trans-

mitted on AD₀-AD₇, regardless of the I/O address. (AD₈-AD₁₅ contain arbitrary data in this case.) For an I/O transaction, the address indicates a peripheral and a particular register or function within that peripheral.

Null Transactions

The two kinds of null transactions are distinguished by the Status lines: internal operation and memory refresh. Both transactions look like a memory read transaction except that Data Strobe remains High and no data is transferred.

For an internal operation transaction, the Address lines contain arbitrary data when Address Strobe goes High. This transaction is initiated to maintain a minimum transaction rate when a bus master is doing a long internal

operation (to support memories which generate refresh cycles from Address Strobe).

For a memory refresh transaction, the Address lines contain a refresh address when Address Strobe goes High. This transaction is used to refresh a row of a dynamic memory.

Any memory or I/O transaction can be suppressed (effectively turning it into a null transaction) by keeping Data Strobe High throughout the transaction.

Interrupts

A complete interrupt cycle consists of an interrupt request followed by an interrupt-acknowledge transaction. The request, which consists of $\overline{INT}$ pulled Low by a peripheral, notifies the CPU that an interrupt is pending. The interrupt-acknowledge transaction, which is initiated by the CPU as a result of the request, performs two functions: it selects the peripheral whose interrupt is to be acknowledged, and it obtains a vector that identifies the selected device and cause of interrupt.

A peripheral can have one or more sources of interrupt. Each interrupt source has three

bits that control how it generates interrupts. These bits are an Interrupt Pending bit (IP), and Interrupt Enable bit (IE), and an Interrupt Under Service bit (IUS).

A peripheral may also have one or more vectors for identifying the source of an interrupt during an interrupt-acknowledge transaction. Each interrupt source is associated with one interrupt vector and each interrupt vector can have one or more interrupt sources associated with it. Each vector has a Vector Includes Status bit (VIS) controlling its use.

Finally, each peripheral has three bits for

controlling interrupt behavior for the whole device. These are a Master Interrupt Enable bit (MIE), a Disable Lower Chain bit (DLC), and a No Vector bit (NV).

Peripherals are connected together via an interrupt daisy chain formed with their IEI and IEO pins (Figure 4). The interrupt sources within a device are similarly connected into this chain with the overall effect being a daisy chain connecting the interrupt sources. The daisy chain has two functions: during an interrupt-acknowledge transaction it determines which interrupt source is being acknowledged; at all other times it determines which interrupt sources can initiate an interrupt request.

Figure 5 is a state diagram for interrupt processing for an interrupt source (assuming its IE bit is 1). An interrupt source with an interrupt pending (IP = 1) makes an interrupt request (by pulling $\overline{\text{INT}}$ Low) if, and only if, it is enabled (IE = 1, MIE = 1), it does not have an interrupt under service (IUS = 0), no higher priority interrupt is being serviced (IEI = High), and no interrupt-acknowledge transaction is in progress (as indicated by $\overline{\text{INTACK}}$ at the last rising edge of $\overline{\text{AS}}$). IEO is not pulled down by the interrupt source at this time; IEO continues to follow IEI until an interrupt-acknowledge transaction occurs.

Some time after $\overline{\text{INT}}$ has been pulled Low,

the CPU initiates an interrupt-acknowledge transaction. Between the rising edge of $\overline{\text{AS}}$ and the falling edge of $\overline{\text{DS}}$, the IEI/IEO daisy chain settles. Any interrupt source with an interrupt pending (IP = 1) or under service (IUS = 1) holds its IEO line Low; all other interrupt sources make IEO follow IEI. When $\overline{\text{DS}}$ falls, only the highest priority interrupt source with a pending interrupt (IP = 1) has its IEI input High, its IE bit set to 1, and its IUS bit set to 0. This is the interrupt source being acknowledged, and at this point it sets its IUS bit to 1, and, if the peripheral's NV bit is 0, identifies itself by placing the vector on $\text{AD}_0\text{--AD}_7$. If the NV bit is 1, then the peripheral's $\text{AD}_0\text{--AD}_7$ pins remain floating, thus allowing external circuitry to supply the vector. (All interrupts, including the Z8000's non-vectorized interrupt, need a vector for identifying the source of an interrupt.) If the vector's VIS bit is 1, the vector will also contain status information further identifying the source of the interrupt. If the VIS bit is 0, the vector held in the peripheral will be output without modification.

While an interrupt source has an interrupt under service (IUS = 1), it prevents all lower priority interrupt sources from requesting interrupts by forcing IEO Low. When interrupt servicing is complete, the CPU must reset the IUS bit and, in most cases, the IP bit (by means of an I/O transaction).

2

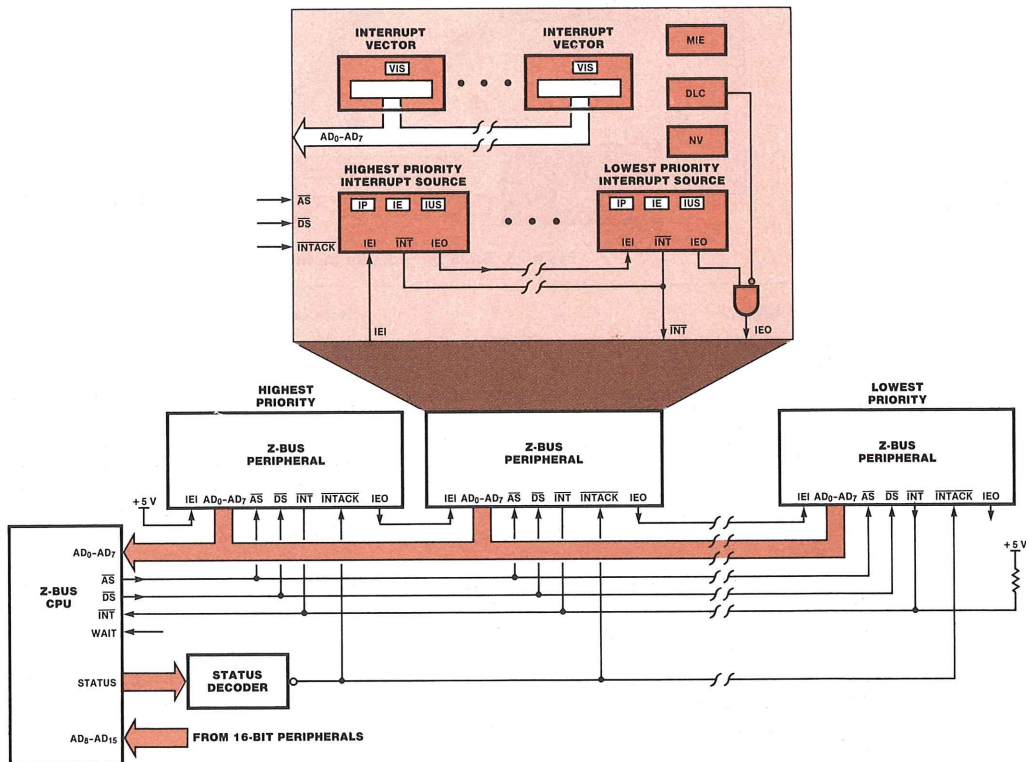


Figure 4. Interrupt Connections

A peripheral's Master Interrupt Enable bit (MIE) and Disable Lower Chain bit (DLC) can modify the behavior of the peripheral's interrupt sources in the following way: if the MIE bit is 0, the effect is as if every Interrupt Enable bit (IE) in the peripheral were 0; thus all interrupts from the peripheral are disabled. If the DLC bit is 1, the effect is to force the peripheral's IEO output Low, thus disabling all lower priority devices from initiating interrupt requests.

Polling can be done by disabling interrupts (using MIE and DLC) and by reading peripherals to detect pending interrupts. Each Z-Bus peripheral has a single directly addressable register that can be read to determine if there is an interrupt pending in the device and, if so, what interrupt source it is from.

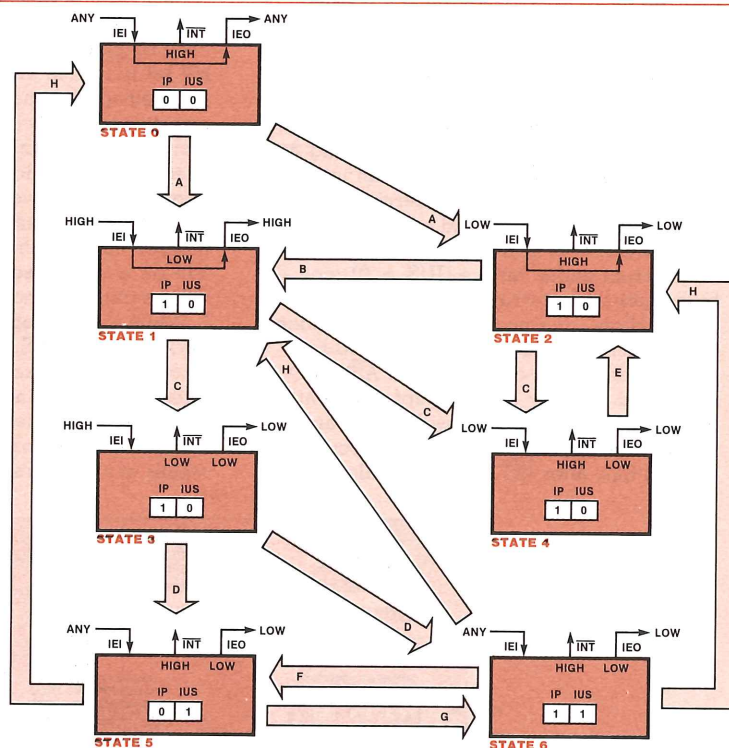


Figure 5. State Diagram for an Interrupt Source

Transition Legend

- A** The peripheral detects an interrupt condition and sets Interrupt Pending.
- B** All higher priority peripherals finish interrupt service, thus allowing IEO to go High.
- C** An interrupt-acknowledge transaction starts, and the IEO/IEO daisy chain settles.
- D** The interrupt-acknowledge transaction terminates with the peripheral selected. Interrupt Under Service (IUS) is set to 1, and Interrupt Pending (IP) may or may not be reset.
- E** The interrupt-acknowledge transaction terminates with a higher priority device having been selected.
- F** The Interrupt Pending bit in the peripheral is reset by an I/O operation.
- G** A new interrupt condition is detected by the peripheral, causing IP to be set again.
- H** Interrupt service is terminated for the peripheral by resetting IUS.

State Legend

- 0** No interrupts are pending or under service for this peripheral.
- 1** An interrupt is pending, and an interrupt request has been made by pulling INT Low.
- 2** An interrupt is pending, but no interrupt request has been made because a higher priority peripheral has an interrupt under service, and this has forced IEO Low.
- 3** An interrupt-acknowledge sequence is in progress, and no higher priority peripheral has a pending interrupt.
- 4** An interrupt-acknowledge sequence is in progress, but a higher priority peripheral has a pending interrupt, forcing IEO Low.
- 5** The peripheral has an interrupt under service. Service may be temporarily suspended (indicated by IEO going Low) if a higher priority device generates an interrupt.
- 6** This is the same as State 5 except that an interrupt is also pending in the peripheral.

**Bus
Requests**

To generate transactions on the bus, a bus requester must gain control of the bus by making a bus request. This is done by forcing $\overline{\text{BUSREQ}}$ Low (Figure 6). A bus request can be made only if $\overline{\text{BUSREQ}}$ is initially High, indicating that the bus is controlled by the CPU and no other device is requesting it.

After $\overline{\text{BUSREQ}}$ is pulled Low, the Z-Bus CPU relinquishes the bus and indicates this condition by making $\overline{\text{BUSACK}}$ Low. The Low on $\overline{\text{BUSACK}}$ is propagated through the $\overline{\text{BAI}}/\overline{\text{BAO}}$ daisy chain (Figure 6). $\overline{\text{BAI}}$ follows $\overline{\text{BAO}}$ for components not requesting the bus, and any component requesting the bus holds its

$\overline{\text{BAO}}$ High, thereby locking out all lower priority users.

A bus requester gains control of the bus when its $\overline{\text{BAI}}$ input goes Low. When it is ready to relinquish the bus, it stops pulling $\overline{\text{BUSREQ}}$ Low and allows $\overline{\text{BAO}}$ to follow $\overline{\text{BAI}}$. This permits lower priority devices that made simultaneous requests to gain control of the bus. When all simultaneously requesting devices have relinquished the bus, $\overline{\text{BUSREQ}}$ goes High, returning control of the bus to the CPU and allowing other devices to request it.

The protocol to be followed in making a bus request is shown in Figure 7.

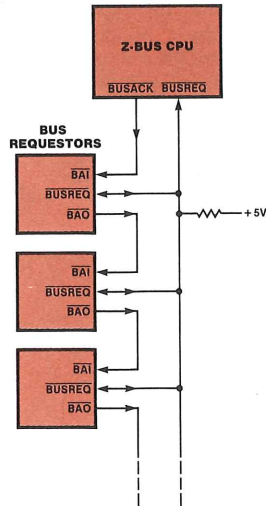


Figure 6. Bus Request Connections

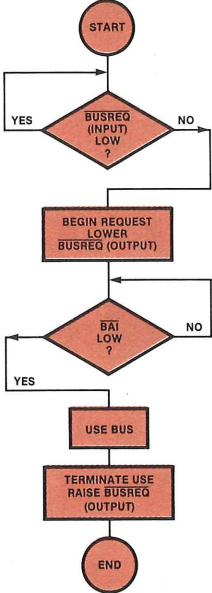


Figure 7. Bus Request Protocol

**Resource
Requests**

Resource requests are used to obtain control of a resource that is shared between several users. The resource can be a common bus, a common memory or any other resource. The requestor can be any component capable of implementing the request protocol.

Unlike the Z-Bus itself, no component has control of a general resource by default; every device must acquire the resource before using it. All devices sharing the general resource drive the $\overline{\text{MMRQ}}$ line (Figure 8). When Low, the $\overline{\text{MMRQ}}$ line indicates that the resource is being acquired or used by some device. The $\overline{\text{MMST}}$ pin allows each device to observe the state of the $\overline{\text{MMRQ}}$ line.

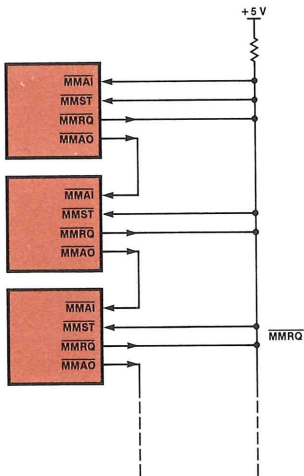


Figure 8. Resource Request Connections

Resource Request (Continued)

When $\overline{\text{MMRQ}}$ is High, a device may initiate a resource request by pulling $\overline{\text{MMRQ}}$ Low (Figure 9). The resulting Low on $\overline{\text{MMRQ}}$ is propagated through the $\overline{\text{MMAI}}/\overline{\text{MMAO}}$ daisy chain. If a device is not requesting the resource, its $\overline{\text{MMAO}}$ output follows its $\overline{\text{MMAI}}$ input. Any device making a resource request forces its $\overline{\text{MMAO}}$ output High to deny use of the resource to lower priority devices.

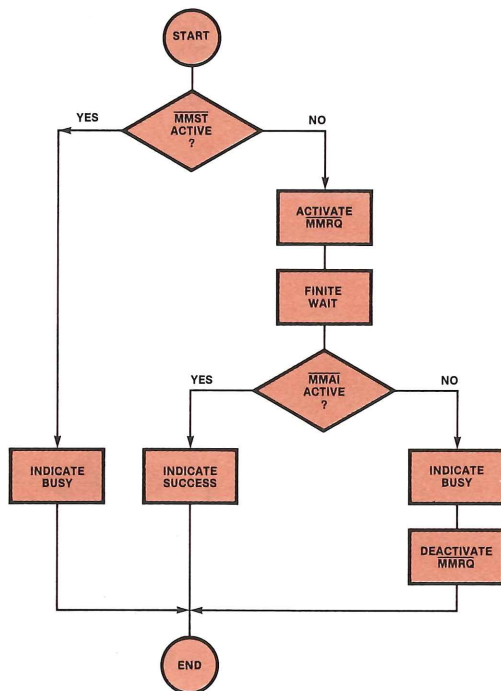


Figure 9. Resource Request Protocol

A device gains control of the resource if its $\overline{\text{MMAI}}$ input is Low (and its $\overline{\text{MMAO}}$ output is High) after a sufficient delay to let the daisy chain settle. If the device does not obtain the resource after this short delay, it must stop pulling $\overline{\text{MMRQ}}$ Low and make another request at some later time when $\overline{\text{MMRQ}}$ is again High. When a device that has gained control of a resource is finished, it releases the resource by allowing $\overline{\text{MMRQ}}$ to go High.

The four unidirectional lines of the resource request chain allow the use of line drivers, thus facilitating connection of components separated by some distance. In the case of the Z8000 CPU, the four resource request lines may be mapped into the CPU $\overline{\text{MI}}$ and $\overline{\text{MO}}$ pins using the logic shown in Figure 10. With this configuration, the Multi-Micro Request Instruction (MREQ) performs a resource request.

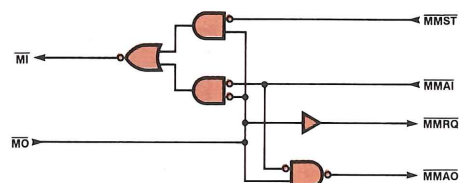
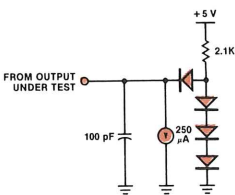


Figure 10. Bus Request Logic for Z8000

Test Conditions

The timing characteristics given in this document reference 2.0 V as High and 0.8 V as Low. The following test load circuit is assumed. The effect of larger capacitive loadings can be calculated by delaying output signal transitions by 10 ns for each additional 50 pF of load up to a maximum 200 pF.



DC Characteristics

The following table states the dc characteristics for the input and output pins of Z-Bus

components. All voltages are relative to ground.

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC} + .3$	V	
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = 2.0\text{mA}$
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = 250\mu\text{A}$
I_{IL}	Input Leakage Current		± 10	μA	$V_{IN} = .4 \text{ to } 2.4 \text{ V}$
I_{OL}	3-State Output Leakage Current in Float		± 10	μA	$V_{OUT} = .4 \text{ to } 2.4 \text{ V}$

Capacitance

The following table gives maximum pin capacitance for Z-Bus components. Capacitance is specified at a frequency of 1 MHz over the temperature range of the component. Unused pins are returned to ground.

Symbol	Parameter	Max (pF)
C_{IN}	Input Capacitance	5
C_{OUT}	Output Capacitance	7
$C_{I/O}$	Bidirectional Capacitance	8

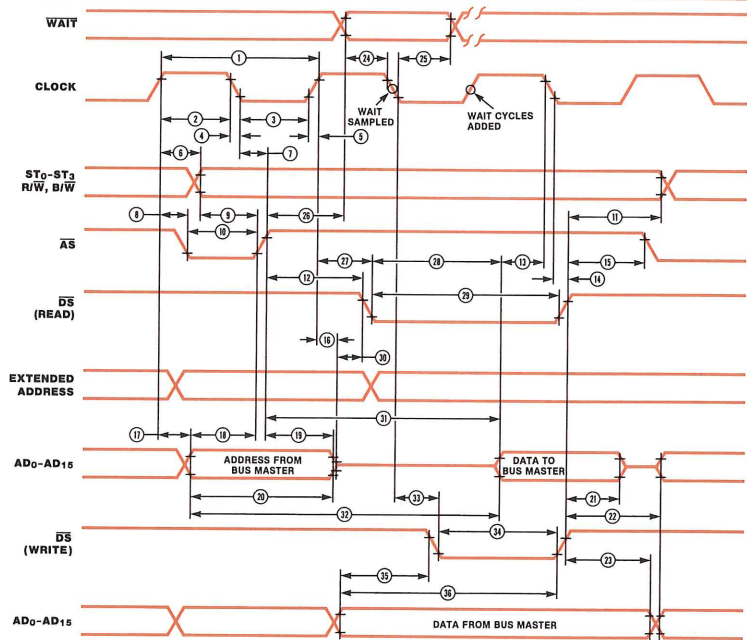
Timing Diagrams

The following diagrams and tables give the timing for each kind of transaction (except null transactions). Timings are given separately for bus masters and for peripherals and memories and are intended to give the minimum timing requirements which a Z-Bus component must meet. An individual component will have more detailed and sometimes more stringent timing specifications. The differences between bus master timing and peripheral and memory timing allow for buffer and decoding circuit

delays and for signal skew. The timing given for memories is a constraint on bus-compatible memories (like the Z6132 Quasi-Static RAM) and is not intended to constrain memory subsystems constructed from conventional components.

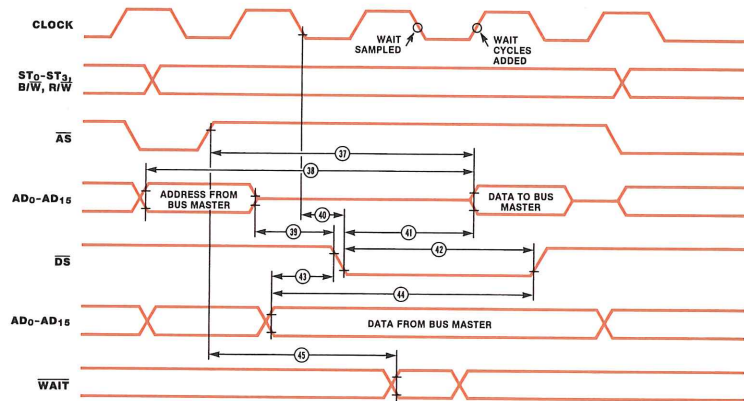
Besides these timings, there is a requirement that at least 128 transactions be initiated in any 2 ms period. This accommodates memories that generate refresh cycles from Address Strobe.

Bus Master Timing

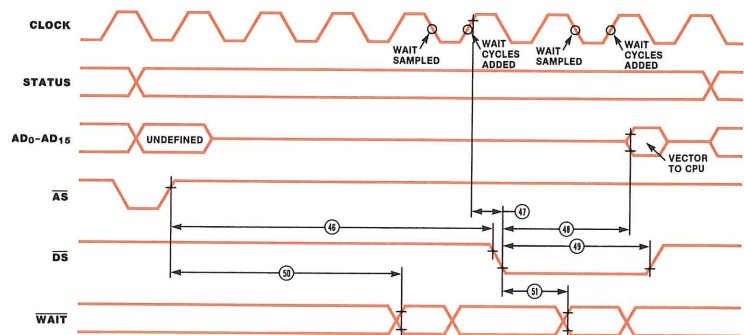


Parameters 1-25 are common to all transactions.

I/O Transaction Timing



Interrupt Acknowledge Timing

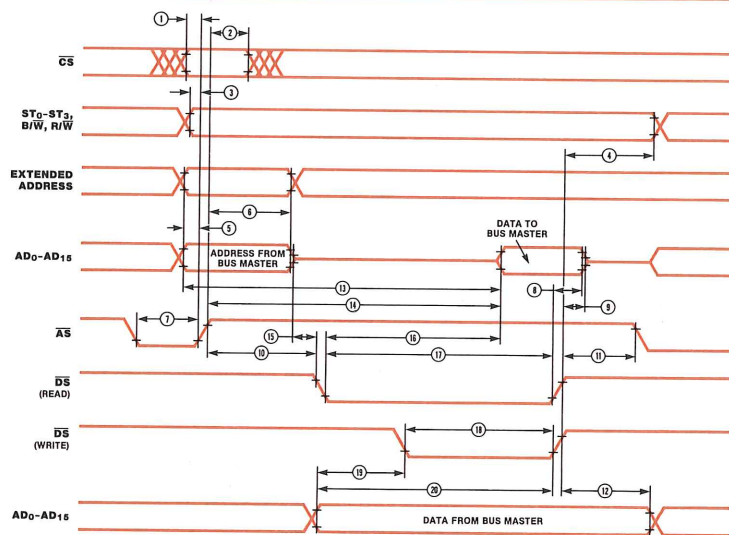


**Bus Master
Timing
Parameters**

Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes
All Transactions					
1	TpC	Clock Period	250	2000	
2	TwCh	Clock High Width	105	1895	
3	TwCl	Clock Low Width	105	1895	
4	TfC	Clock Fall Time		20	
5	TrC	Clock Rise Time		20	
6	TdC(S)	Clock ↑ To Status Valid Delay		100	
7	TdC(ASr)	Clock ↑ To $\overline{AS}$ ↑ Delay		90	
8	TdC(ASf)	Clock ↑ To $\overline{AS}$ ↓ Delay		80	
9	TdS(AS)	Status Valid To $\overline{AS}$ ↑ Delay	40		
10	TwAS	$\overline{AS}$ Low Width	80		
11	TdDS(S)	$\overline{DS}$ ↑ To Status Not Valid Delay	80		
12	TdAS(DS)	$\overline{AS}$ ↑ To $\overline{DS}$ ↓ Delay	70	2095	3
13	TsDR(C)	Read Data To Clock ↓ Setup Time	50		
14	TdC(DS)	Clock ↓ To $\overline{DS}$ ↑ Delay		70	
15	TdDS(AS)	$\overline{DS}$ ↑ To $\overline{AS}$ ↓ Delay	70		
16	TdC(Az)	Clock ↑ To Address Float Delay		65	
17	TdC(A)	Clock ↑ To Address Valid Delay		90	
18	TdA(AS)	Address Valid To $\overline{AS}$ ↑ Delay	50		1
19	TdAS(A)	$\overline{AS}$ ↑ To Address Not Valid Delay	60		1
20	TwA	Address Valid Width	150		
21	ThDR(DS)	Read Data To $\overline{DS}$ ↑ Hold Time	0		
22	TdDS(A)	$\overline{DS}$ ↑ To Address Active Delay	80		
23	TdDS(DW)	$\overline{DS}$ ↑ To Write Data Not Valid Delay	80		
24	TsW(C)	WAIT To Clock ↓ Setup Time	50		2
25	ThW(C)	WAIT To Clock ↓ Hold Time	0		2
Memory Transactions					
26	TdAS(W)	$\overline{AS}$ ↑ To WAIT Required Valid		90	
27	TdC(DSR)	Clock ↓ To $\overline{DS}$ (Read) ↓ Delay		120	
28	TdDSR(DR)	$\overline{DS}$ (Read) ↓ To Read Data Required Valid		155	
29	TwDSR	$\overline{DS}$ (Read) Low Width		250	
30	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		
31	TdAS(DR)	$\overline{AS}$ ↑ To Read Data Required Valid		290	
32	TdA(DR)	Address Valid To Read Data Required Valid		400	
33	TdC(DSW)	Clock ↓ To $\overline{DS}$ (Write) ↓ Delay		95	
34	TwDSW	$\overline{DS}$ (Write) Low Width	160		
35	TdDW(DSWf)	Write Data Valid To $\overline{DS}$ (Write) ↓ Delay	50		
36	TdDW(DSWr)	Write Data Valid To $\overline{DS}$ (Write) ↑ Delay	230		
I/O Transactions					
37	TdAS(DR)	$\overline{AS}$ ↑ To Read Data Required Valid		540	
38	TdA(DR)	Address Valid To Read Data Required Valid		650	
39	TdAz(DSI)	Address Float To $\overline{DS}$ (I/O) ↓	0		
40	TdC(DSI)	Clock ↓ To $\overline{DS}$ (I/O) ↓		120	
41	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ To Read Data Required Valid		315	
42	TwDSI	$\overline{DS}$ (I/O) Low Width	400		
43	TdDW(DSI f)	Write Data To $\overline{DS}$ (I/O) ↓ Delay	50		
44	TdDW(DSI r)	Write Data To $\overline{DS}$ (I/O) ↑ Delay	480		
45	TdAS(W)	$\overline{AS}$ ↑ To WAIT Required Valid		340	
Interrupt-Acknowledge Transactions					
46	TdAS(DSA)	$\overline{AS}$ ↑ To $\overline{DS}$ (Acknowledge) ↓ Delay	960		
47	TdC(DSA)	Clock ↑ To $\overline{DS}$ (Acknowledge) ↓ Delay		120	
48	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ To Read Data Required Valid		420	
49	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	485		
50	TdAS(W)	$\overline{AS}$ ↑ To Wait Required Valid		840	
51	TdDSA(W)	$\overline{DS}$ (Acknowledge) ↓ To Wait Required Valid		130	

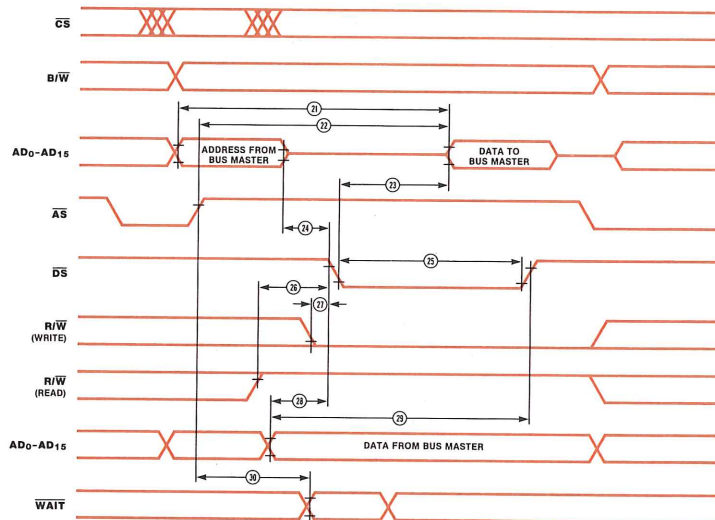
1. Timing for extended addresses is CPU dependent; however, extended addresses must be valid at least as soon as addresses are valid on AD₀-AD₁₅ and must remain valid at least as long as addresses are valid on AD₀-AD₁₅.
2. The exact clock cycle that wait is sampled on depends on the type of transaction; however, wait always has the given setup and hold times to the clock.
3. The maximum value for TdAS(DS) does not apply to Interrupt-Acknowledge Transactions.

Memory and Peripheral Timing

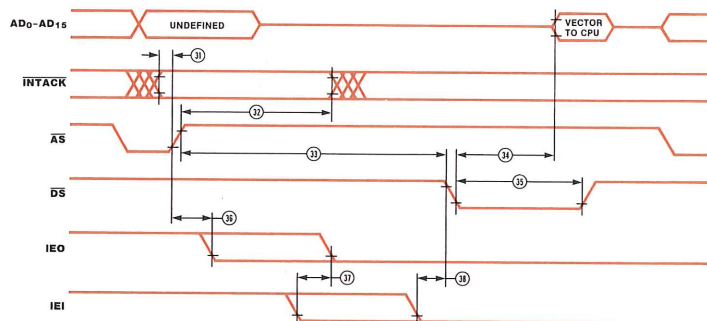


Parameters 1-12 are common to all transactions.

I/O Transaction Timing



Interrupt Acknowledge Timing



Memory and Peripheral Timing Parameters	Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes
All Transactions						
	1	TsCS(AS)	$\overline{CS}$ To $\overline{AS}$ ↑ Setup Time	0		1
	2	ThCS(AS)	$\overline{CS}$ To $\overline{AS}$ ↑ Hold Time	60		1
	3	TsS(AS)	Status To $\overline{AS}$ ↑ Setup Time	20		2
	4	ThS(DS)	Status To $\overline{DS}$ ↑ Hold Time	60		
	5	TsA(AS)	Address To $\overline{AS}$ ↑ Setup Time	10		1
	6	ThA(AS)	Address To $\overline{AS}$ ↑ Hold Time	50		1
	7	TwAS	$\overline{AS}$ Low Width	70		
	8	TdDS(DR)	$\overline{DS}$ ↑ To Read Data Not Valid Delay	0		
	9	TdDS(DRz)	$\overline{DS}$ ↑ To Read Data Float Delay		70	
	10	TdAS(DS)	$\overline{AS}$ ↑ To $\overline{DS}$ ↓ Delay	60	2095	5
	11	TdDS(AS)	$\overline{DS}$ ↑ To $\overline{AS}$ ↓ Delay	50		
	12	ThDW(DS)	Write Data To $\overline{DS}$ ↑ Hold Time	30		1
Memory Transactions						
	13	TdA(DR)	Address Required Valid To Read Data Valid Delay		340	
	14	TdAS(DR)	$\overline{AS}$ ↑ To Read Data Valid Delay		230	
	15	TdAz(DSR)	Address Float To $\overline{DS}$ (Read) ↓ Delay	0		
	16	TdDSR(DR)	$\overline{DS}$ (Read) ↓ To Read Data Valid Delay		95	
	17	TwDSR	$\overline{DS}$ (Read) Low Width	240		
	18	TwDSW	$\overline{DS}$ (Write) Low Width	150		
	19	TsDW(DSWf)	Write Data To $\overline{DS}$ (Write) ↓ Setup Time	30		
	20	TsDW(DSWr)	Write Data To $\overline{DS}$ (Write) ↓ Setup Time	210		
I/O Transactions						
	21	TdA(DR)	Address Required Valid To Read Data Valid Delay		590	
	22	TdAS(DR)	$\overline{AS}$ ↑ To Read Data Valid Delay		480	
	23	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ To Read Data Valid Delay		255	
	24	TdAz(DSI)	Address Float To $\overline{DS}$ (I/O) ↓ Delay	0		
	25	TwDSI	$\overline{DS}$ (I/O) Low Width	390		
	26	TsRWR(DSI)	R/ $\overline{W}$ (Read) To $\overline{DS}$ (I/O) ↓ Setup Time	100		
	27	TsRWW(DSI)	R/ $\overline{W}$ (Write) To $\overline{DS}$ (I/O) ↓ Setup Time	0		
	28	TsDW(DSIr)	Write Data To $\overline{DS}$ (I/O) ↓ Setup Time	30		
	29	TsDW(DSIr)	Write Data To $\overline{DS}$ (I/O) ↓ Setup Time	460		
	30	TdAS(W)	$\overline{AS}$ ↑ To WAIT Valid Delay	195		
Interrupt-Acknowledge Transactions						
	31	TsIA(AS)	$\overline{INTACK}$ To $\overline{AS}$ ↑ Setup Time	0		
	32	ThIA(AS)	$\overline{INTACK}$ To $\overline{AS}$ ↑ Hold Time	250		
	33	TdAS(DSA)	$\overline{AS}$ ↑ To $\overline{DS}$ (Acknowledge) ↓ Delay	940		
	34	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ To Read Data Valid Delay	360		
	35	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	475		
	36	TdAS(IEO)	$\overline{AS}$ ↑ To IEO ↓ Delay			3, 4
	37	TdIEIf(IEO)	IEI ↓ To IEO ↓ Delay			4
	38	TsIEI(DSA)	IEI To $\overline{DS}$ (Acknowledge) ↓ Setup Time			4

1. Parameter does not apply to Interrupt Acknowledge Transactions
2. Does not cover R/ $\overline{W}$ for I/O Transactions
3. Applies only to a peripheral which is pulling $\overline{INT}$ Low at the beginning of the Interrupt-Acknowledge Transaction.
4. These parameters are device dependent. The parameters for the devices in any particular daisy chain must meet the following constraint: for any two peripherals in the daisy chain, TdAS(DSA) must be greater than the sum of TdAS(IEO) for the higher priority peripheral, TsIEI(DSA) for the lower priority peripheral, and TdIEH(IEO) for each peripheral separating them in the daisy chain.
5. The maximum value for TdAS(DS) does not apply to Interrupt-Acknowledge Transactions.

Z8010 MMU Memory Management Unit



NEW
1980

Product Specification

Preliminary

February 1980

Features

- Dynamic segment relocation makes software addresses independent of physical memory addresses.
- Sophisticated memory-management features include access validation that protects memory areas from unauthorized or unintentional access, and a write-warning indicator that predicts stack overflow.
- 64 variable-sized segments from 256 to

65,536 bytes can be mapped into a total physical address space of 16M bytes; all 64 segments are randomly accessible.

- Multiple MMUs can support several translation tables for each Z8001 address space.
- MMU architecture supports multi-programming systems and virtual memory implementations.

2

General Description

The Z8010 Memory Management Unit (MMU) manages the large 8M byte addressing spaces of the Z8001 CPU. The MMU provides dynamic segment relocation as well as numerous memory protection features.

Dynamic segment relocation makes user software addresses independent of the physical memory addresses, thereby freeing the user from specifying where information is actually

located in the physical memory. It also provides a flexible, efficient method for supporting multi-programming systems. The MMU uses a translation table to transform the 23-bit logical address output from the Z8001 CPU into a 24-bit address for the physical memory. (Only logical memory addresses go to an MMU for translation; I/O addresses and data, in general, must bypass this component.)

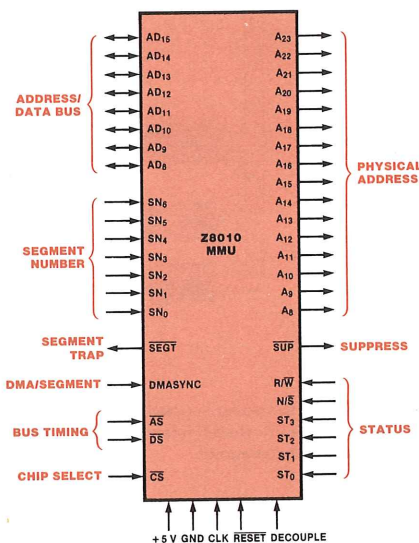


Figure 1. Pin Functions

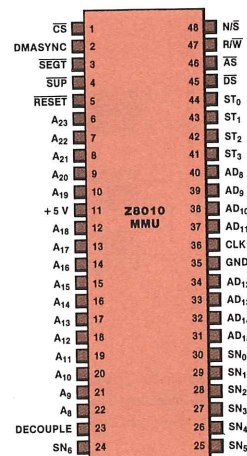


Figure 2. Pin Assignments

**General
Description**
(Continued)

Memory segments are variable in size from 256 bytes to 64K bytes, in increments of 256 bytes. Pairs of MMUs support the 128 segment numbers available for the various Z8001 CPU address spaces. Within an address space, any number of MMUs can be used to accommodate multiple translation tables for System and Normal operating modes, or to support more sophisticated memory-management systems.

MMU memory-protection features safeguard memory areas from unauthorized or unintended access by associating special access restrictions with each segment. A segment is assigned a number of attributes when its descriptor initially entered into the MMU. When a memory reference is made, these attributes are checked against the status information supplied by the Z8001 CPU. If a

mismatch occurs, a trap is generated and the CPU is interrupted. The CPU can then check the status registers of the MMU to determine the cause.

Segments are protected by modes of permitted use, such as read only, system only, execute only and CPU-access only. Other segment management features include a write-warning zone useful for stack operations and status flags that record read or write accesses to each segment.

The MMU is controlled via 22 Special I/O instructions from the Z8001 CPU in System mode. With these instructions, system software can assign program segments to arbitrary memory locations, restrict the use of segments and monitor whether segments have been read or written.

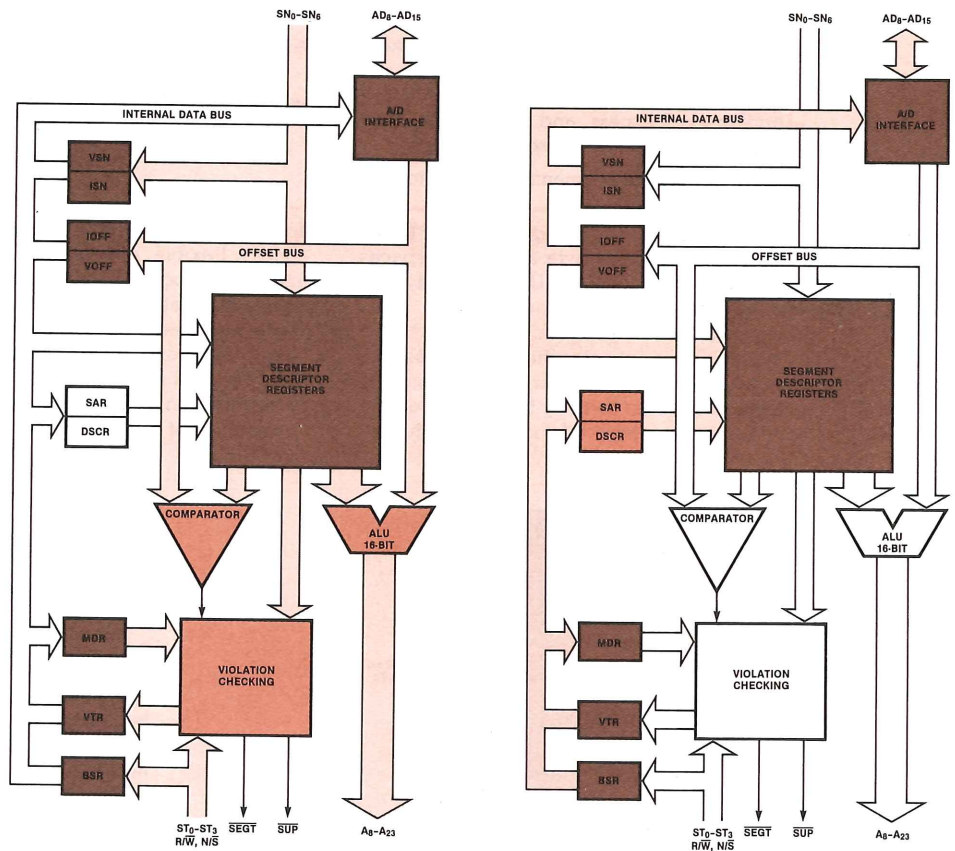


Figure 3. The shaded areas in these block diagrams illustrate the resources used in the two modes of MMU operation. In the Address Translation Mode shown on the left, addresses are translated automatically. In the Command Mode shown on the right, specific registers are accessed using Special I/O commands.

Segmented Addressing

A segmented addressing space—compared with linear addressing—is closer to the way a programmer uses memory because each procedure and data set can reside in its own segment.

The 8M byte Z8001 addressing spaces are divided into 128 relocatable segments of up to 64K bytes each. A 23-bit segmented address uses a 7-bit segment address to point to the segment, and a 16-bit offset to address any byte relative to the beginning of the segment. The two parts of the segmented address may be manipulated separately.

The MMU divides the physical memory into 256-byte blocks. Segments consist of physically contiguous blocks. Certain segments may be designated so that writes into the last block generate a warning trap. If such a segment is used as a stack, this warning can be used to increase the segment size and prevent a stack overflow error.

The addresses manipulated by the programmer, used by instructions and output by the Z8001 are called *logical addresses*. The MMU takes the logical addresses and transforms them into the *physical addresses* required for accessing the memory (Figure 4). This address transformation process is called *relocation*.

The relocation process is transparent to user software. A translation table in the MMU associates the 7-bit segment number with the base address of the physical memory segment.

Memory Protection

Each memory segment is assigned several attributes that are used to provide memory access protection. A memory request from the Z8001 CPU is accompanied by status information that indicates the attributes of the memory request. The MMU compares the memory request attributes with the segment attributes and generates a Trap Request whenever it detects an attribute violation. Trap Request informs the Z8001 CPU and the system control program of the violation so that appropriate action can be taken to recover. The MMU also generates the Suppress signal $\overline{SUP}$ in the event of an access violation. Suppress can be used by a memory system to inhibit stores into the memory and thus protect the contents of the memory from erroneous changes.

Five attributes can be associated with each segment. When an attempted access violates any one of the attributes associated with a segment, a Trap Request and a Suppress signal are generated by the MMU. These attributes are read only, execute only, system access only, inhibit CPU accesses and inhibit DMA accesses.

Segments are specified by a base address

The 16-bit logical address offset is added to the physical base address to obtain the actual physical memory location. Because a base address always has a low byte equal to zero, only the high-order 16 bits are stored in the MMU and used in the addition. Thus the low-order byte of the physical memory location is the same as the low-order byte of the logical address offset. This low-order byte therefore bypasses the MMU, thus reducing the number of pins required.

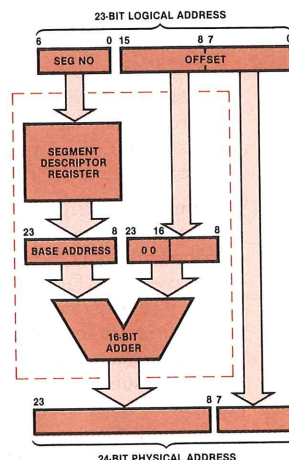


Figure 4. Logical-to-Physical Address Translation

and a range of legal offsets to this base address. On each access to a segment, the offset is checked against this range to insure that the access falls within the allowed range. If an access that lies outside the segment is attempted, Trap Request and Suppress are generated.

Normally the legal range of offsets within a segment is from 0 to $256N + 255$ bytes, where $0 \leq N \leq 255$. However, a segment may be specified so that legal offsets range from $256N$ to $65,535$ bytes, where $0 \leq N \leq 255$. The latter type of segment is useful for stacks since the Z8001 stack manipulation instructions cause stacks to grow toward lower memory locations. Thus when a stack grows to the limit of its allocated segment, additional memory can be allocated on the correct end of the segment. As an aid in maintaining stacks, the MMU detects when a write is performed to the lowest allocated 256 bytes of these segments and generates a Trap Request. No Suppress signal is generated so the write is allowed to proceed. This write warning can then be used to indicate that more memory should be allocated to the segment.

The MMU contains three types of registers: Segment Descriptor, Control and Status. A set of 64 Segment Descriptor Registers supplies the information needed to map logical memory addresses to physical memory locations. The segment number of a logical address determines which Segment Descriptor Register is used in address translation. Each Descriptor Register also contains the necessary information for checking that the segment location referenced is within the bounds of the segment and that the type of reference is permitted. It also indicates whether the segment has been read or written.

In addition to the Segment Descriptor Registers, the Z8010 MMU contains three 8-bit control registers for programming the device and six 8-bit status registers that record information in the event of an access violation.

Segment Descriptor Registers. Each of the 64 Descriptor Registers contains a 16-bit base address field, an 8-bit limit field and an 8-bit attribute field (Figure 5). The base address field is subdivided into high- and low-order bytes that are loaded one byte at a time when the descriptor is initialized. The limit field contains a value N that indicates N + 1 blocks of 256 bytes have been allocated to the segment.*

The attribute field contains eight flags (Figure 6). Five are related to protecting the segment against certain types of access, one indicates the special structure of the segment, and two encode the types of accesses that have been made to the segment. A flag is set when its value is 1. The following brief descriptions indicate how these flags are used.

Read-Only (RD). When this flag is set, the segment is read only and is protected against any write access.

System-Only (SYS). When this flag is set, the segment can be accessed only in system mode, and is protected against any access in normal mode.

CPU-Inhibit (CPUI). When this flag is set, the segment is not accessible to the currently executing process, and is protected against any memory access by the CPU. The segment is, however, accessible under DMA.

Execute-Only (EXC). When this flag is set, the segment can be accessed only during an instruction fetch cycle, and thus is protected against any access during other cycles.

DMA-Inhibit (DMAI). When this flag is set, the segment can be accessed only by the CPU, and thus is protected against any access under DMA.

Direction and Warning (DIRW). When this flag is set, the segment memory locations are considered to be organized in descending order and each write to the segment is checked for access to the last 256-byte block. Such an access generates a trap to warn of potential segment overflow, but no Suppress signal is generated.

Changed (CHG). When this flag is set, the segment has been changed (written). This bit is set automatically during any write access to this segment if the write access does not cause any violation.

Referenced (REF). When this flag is set, the segment has been referenced (either read or written). This bit is set automatically during any access to the segment if the access does not cause a violation.

*In the stack mode, segment size is 64K-256N.

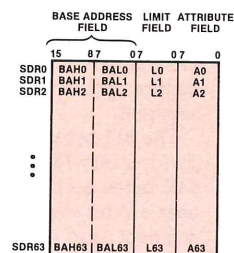


Figure 5. Segment Descriptor Registers

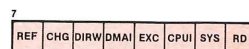


Figure 6. Attribute Field in Segment Descriptor Register

Control Registers. The three user-accessible 8-bit control registers in the MMU direct the functioning of the MMU (Figure 7). The Mode Register provides a sophisticated method for selectively enabling MMUs in multiple-MMU configurations. The Segment Address Register (SAR) selects a particular Segment Descriptor Register to be accessed during a control operation. The Descriptor Selection Counter Register points to a byte within the Segment Descriptor Register to be accessed during a control operation.

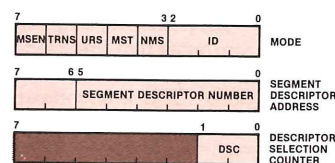


Figure 7. Control Registers

The Mode Register contains a 3-bit identification field (ID) that distinguishes among eight enabled MMUs in a multiple-MMU configuration. This field is used during the segment trap acknowledge sequence (refer to the section on Segment Trap and Acknowledge). In addition, the Mode Register contains five flags.

Multiple Segment Table (MST). This flag indicates whether multiple segment tables are present in the hardware configuration. When this flag is set, more than one table is present and the N/S line must be used to determine whether the MMU contains the appropriate table.

Normal Mode Select (NMS). This flag indicates whether the MMU is to translate addresses when the N/S line is High or Low. If the MST flag is set, the N/S line must match the NMS flag for the MMU to translate segment addresses, otherwise the MMU Address lines remain 3-stated.

Upper Range Select (URS). This flag is used to indicate whether the MMU contains the lower-numbered segment descriptors or the higher-numbered segment descriptors. The most significant bit of the segment number must match the URS flag for the MMU to translate segment addresses, otherwise the MMU Address lines remain 3-stated.

Translate (TRNS). This flag indicates whether the MMU is to translate logical program addresses to physical memory locations or is to pass the logical addresses unchanged to the memory and without protection checking. In the non-translation mode, the most significant byte of the output is the 7-bit segment number and the most significant bit is 0. When this flag is set, the MMU performs address translation and attribute checking.

Master Enable (MSEN). This flag enables or disables the MMU from performing its address translation and memory protection functions. When this flag is set, the MMU performs these tasks; when the flag is clear the Address lines of the MMU remain 3-stated.

The Segment Address Register (SAR) points to one of the 64 segment descriptors. Control commands to the MMU that access segment descriptors implicitly use this pointer to select one of the descriptors. This register has an auto-incrementing capability so that multiple descriptors can be accessed in a block read/write fashion.

The Descriptor Selection Counter Register holds a 2-bit counter that indicates which byte in the descriptor is being accessed during the reading or writing operation. A value of zero in this counter indicates the high-order byte of the base address field is to be accessed, one indicates the low-order byte of the base address, two indicates the limit field and three indicates the attribute field.

Status Registers. Six 8-bit registers contain information useful in recovering from memory access violations (Figure 8). The Violation Type Register describes the conditions that generated the trap. The Violation Segment Number and Violation Offset Registers record the most-significant 15 bits of the logical address that causes a trap. The Instruction Segment Number and Offset Registers record the most-significant 15 bits of the logical address of the last instruction fetched before the first accessing violation. These two registers can be used in conjunction with external circuitry that records the low-order offset byte. At the time of the addressing violation, the Bus Cycle Status Register records the bus cycle status (status code, read/write mode and normal/system mode).

The MMU generates a Trap Request for two general reasons: either it detects an access

violation, such as an attempt to write into a read-only segment, or it detects a warning condition, which is a write into the lowest 256 bytes of a segment with the DIRW flag set. When a violation or warning condition is detected, the MMU generates a Trap Request and automatically sets the appropriate flags. The eight flags in the Violation Type Register describe the cause of a trap.

Read-Only Violation (RDV). Set when the CPU attempts to access a read-only segment and the R/W line is Low.

System Violation (SYSV). Set when the CPU accesses a system-only segment and the N/S line is High.

CPU-Inhibit Violation (CPUIV). Set when the CPU attempts to access a segment with the CPU-inhibit flag set.

Execute-Only Violation (EXCV). Set when the CPU attempts to access an execute-only segment in other than an instruction fetch cycle.

Segment Length Violation (SLV). Set when an offset falls outside of the legal range of a segment.

Primary Write Warning (PWW). Set when an access is made to the lowest 256 bytes of a segment with the DIRW flag set.

Secondary Write Warning (SWW). Set when the CPU pushes data into the last 256 bytes of a system stack and EXCV, CPUIV, SLV, SYSV, RDV or PWW is set. Once this flag is set, subsequent write warnings for accessing the system stack do not generate a Segment Trap request.

Fatal Condition (FATL). Set when any other flag in the Violation Type Register is set and either a violation is detected or a write warning condition occurs in normal mode. This flag is not set during a stack push in system mode that results in a warning condition. This flag indicates a memory access error has occurred in the trap processing routine. Once set, no Trap Request signals are generated on subsequent violations. However, Suppress signals are generated on this and subsequent CPU violations until the FATL flag has been reset.

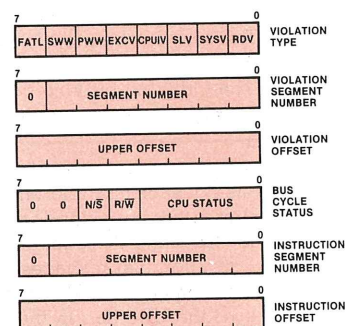


Figure 8. Status Registers

**Segment
Trap and
Acknowledge**

The Z8010 MMU generates a Segment Trap when it detects an access violation or a write warning condition. In the case of an access violation, the MMU also activates Suppress, which can be used to inhibit memory writes and to flag special data to be returned on a read access. Segment Trap remains Low until a Trap Acknowledge signal is received. If a CPU-generated violation occurs, Suppress is asserted for that cycle and all subsequent CPU instruction execution cycles until the end of the instruction. Intervening DMA cycles are not suppressed, however, unless they generate a violation. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only—no Segment Trap Requests are ever generated during DMA cycles.

Segment traps to the Z8001 CPU are handled similarly to other types of interrupts. To service a segment trap, the CPU issues a segment trap acknowledge cycle. The acknowledge cycle is always preceded by an instruction fetch cycle that is aborted (the MMU has been designed so that this dummy cycle is ignored). During the acknowledge cycle all enabled MMUs use the Address/Data lines to indicate their status. An MMU that has generated a Segment Trap Request outputs a 1 on the A/D line associated with the number in its ID field; an MMU that has not generated a segment trap request outputs a 0 on its associated A/D line. A/D lines for which no MMU is associated remain 3-stated. During a

segment trap acknowledge cycle, an MMU uses A/D line $8+i$ if its ID field is i .

Following the acknowledge cycle the CPU automatically pushes the Program Status and Program Counter onto the system stack and loads another Program Status and Program Counter from the Program Status Area. The Segment Trap line is reset during the segment trap acknowledge cycle. Suppress is not generated during the stack push. If the store creates a write warning condition, a Segment Trap Request is generated and is serviced at the end of the context swap. The SWW flag is also set. Servicing this second Segment Trap Request also creates a write warning condition, but because the SWW flag is set, no Segment Trap Request is generated. If a violation rather than a write warning occurs during the context swap, the FATL flag is set rather than the SWW flag. Subsequent violations cause Suppress to be asserted but not Segment Trap Request. Without the SWW and FATL flags, trap processing routines that generate memory violations would repeatedly be interrupted and called to process the trap they created.

The CPU routine to process a trap request should first check the FATL flag to determine if a fatal system error has occurred. If not, the SWW flag should be checked to determine if more memory is required for the system stack. Finally, the trap itself should be processed and the Violation Type Register reset.

**Virtual
Memory**

Several features of the MMU can be used in conjunction with external circuitry to support virtual memory for the Z8001. Segment Trap Request can be used to signal the CPU in the event that a segment is not in primary memory. The CPU-Inhibit Flag can be used to indicate whether a segment is in the memory or in

secondary storage. The Changed and Altered Flags in the attribute field for each segment can aid in implementing efficient segment management policies. The Status Registers can be used in recovering from virtual memory access faults.

**Multiple
MMUs**

MMU architecture directly supports two methods for multiple MMU configurations. The first approach extends single-MMU capability for handling 64 segments to a dual-MMU configuration that manages the 128 different segments the Z8001 can address. This scheme uses the URS flag in the Mode Register in connection with the high-order bit of the segment number (SN_6).

The second approach uses several MMUs to implement multiple translation tables. Multiple tables can be used to reduce the time required to switch tasks by assigning separate tables to each task. Multiple translation tables for multi-

task environments can use the Master Enable Flag to enable the appropriate MMUs through software. Multiple translation tables may also be used to extend the physical memory size beyond 16 megabytes by separating system from normal memory and/or program from data memory. The MST and NMS flags in the Mode Register can be used in conjunction with the N/S line to select the MMU that contains the appropriate table. Special external circuitry that monitors the CPU Status lines can manipulate the MMU N/S line to perform this selection.

DMA Operation

Direct memory access operations may occur between Z8001 instruction cycles and can be handled through the MMU. The MMU permits DMA in either the System or Normal mode of operation. For each memory access, the segment attributes are checked and if a violation is detected, Suppress is activated. Unlike a CPU violation that automatically causes Suppress signals to be generated on subsequent memory accesses until the next instruction, DMA violations generate a Suppress only on a per memory access basis.

The DMA device should note the Suppress signal and record sufficient information to enable the system to recover from the access violation. No Segment Trap Request is ever generated during DMA, hence warning conditions are not signaled. Trap Requests are not issued because the CPU cannot acknowledge such a request.

At the start of a DMA cycle, DMASYNC must go Low, indicating to the MMU the beginning of a DMA cycle. A Low DMASYNC inhibits the MMU from using an indeterminate segment number on lines SN₀-SN₆. When the DMA logical memory address is valid, the DMASYNC line must be High on a rising edge of Clock and the MMU then performs its address translation and access protection functions. Upon the release of the bus at the termination of the DMA cycle the DMASYNC line must again be High. After two clock cycles of DMASYNC High, the MMU assumes that the CPU has control of the bus and that subsequent memory references are CPU accesses. The first instruction fetch occurs at least two cycles after the CPU regains control of the bus. During CPU cycles, DMASYNC should always be High.

MMU Commands

The various registers in the MMU can be read and written using Z8001 CPU special I/O commands. These commands have machine cycles that cause the Status lines to indicate an SIO operation is in progress. During these machine cycles the MMU enters command mode. In this mode, the rising edge of the Address Strobe indicates a command is present on the AD₈-AD₁₅. If this command indicates that data is to be written into one of the MMU registers, the data is read from AD₈-AD₁₅ while Data Strobe is Low. If the command indicates that data is to be read from one of the MMU registers, the data is placed on AD₈-AD₁₅ while Data Strobe is Low.

There are ten commands that read or write various fields in the Segment Descriptor Register. The status of the Read/Write line indicates whether the command is a read or a write.

The auto-incrementing feature of the Segment Address Register (SAR) can be used to block load segment descriptors using the repeat forms of the Special I/O instructions. The SAR is autoincremented at the end of the field. In accessing the base field, first the high-order byte is selected and then the low-order byte. The command accessing the entire Descriptor Register references the fields in the order of base address, limit and attribute.

Opcode (Hex)	Instruction
08	Read/Write Base Field
09	Read/Write Limit Field
0A	Read/Write Attribute Field
0B	Read/Write Descriptor (all fields)
0C	Read/Write Base Field; Increment SAR
0D	Read/Write Limit Field; Increment SAR
0E	Read/Write Attribute Field; Increment SAR
0F	Read/Write Descriptor; Increment SAR
15	Set All CPU-Inhibit Attribute Flags
16	Set All DMA-Inhibit Attribute Flags

Three commands are used to read and write the control registers.

Opcode (Hex)	Instruction
00	Read/Write Mode Register
01	Read/Write Segment Address Register
20	Read/Write Descriptor Selector Counter Register

The Status Registers are read-only registers, although the Violation Type Register (VTR) can be reset. Nine instructions access these registers.

Opcode (Hex)	Instruction
02	Read Violation Type Register
03	Read Violation Segment Number Register
04	Read Violation Offset (High-byte) Register
05	Read Bus Status Register
06	Read Instruction Segment Number Register
07	Read Instruction Offset (High-byte) Register
11	Reset Violation Type Register
13	Reset SWW Flag in VTR
14	Reset FATL Flag in VTR

MMU Timing

The Z8010 translates addresses and checks for access violations by stepping through sequences of basic clock cycles corresponding to the cycle structure of the Z8001 CPU. The following timing diagrams show the relative timing relationships of MMU signals during the basic operations of memory read/write and MMU control commands. For exact timing information, refer to the composite timing diagram.

Memory Read and Write. Memory read and instruction fetch cycles are identical, except for the status information on the ST_0-ST_3 inputs. During a memory read cycle (Figure 9) the 7-bit segment number is input on SN_0-SN_6 one clock period earlier than the address offset; a High on $DMASync$ during T_3 indicates that the segment offset data is valid. The most significant eight bits of the address offset are placed on the AD_0-AD_{15} inputs early in the

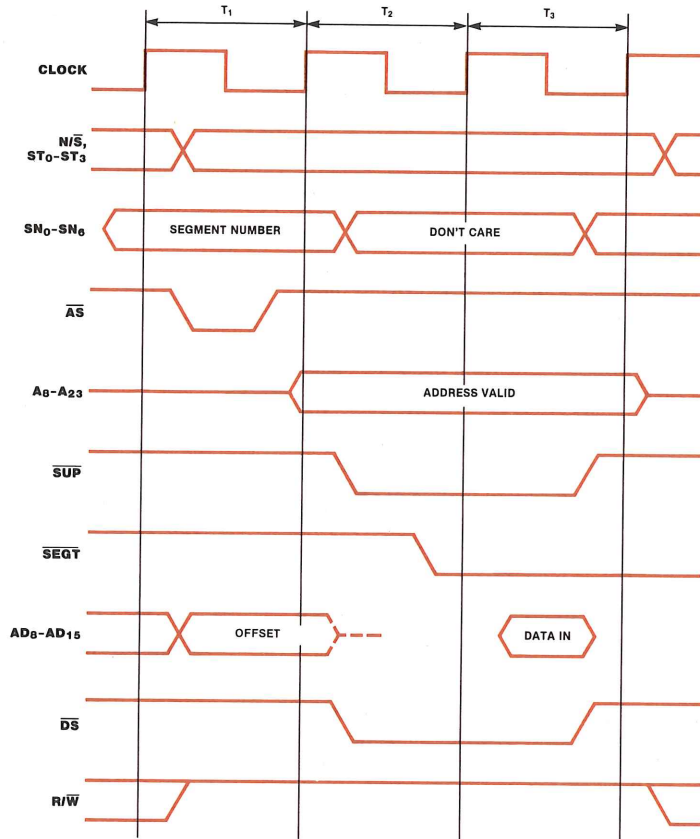


Figure 9. Memory Read Timing

**MMU
Timing**
(Continued)

first clock period. Valid address offset data is indicated by the rising edge of Address Strobe. Status and mode information become valid early in the memory access cycle and remain stable throughout. The most significant 16-bits of the address (physical memory location) remain valid until the end of T₃. Segment Trap Request and Suppress are asserted in T₂.

Segment Trap Request remains Low until Segment Trap Acknowledge is received. Suppress is asserted during the current machine cycle and terminates during T₃. Suppress is repeatedly asserted during CPU instruction execution cycles until the current instruction has terminated.

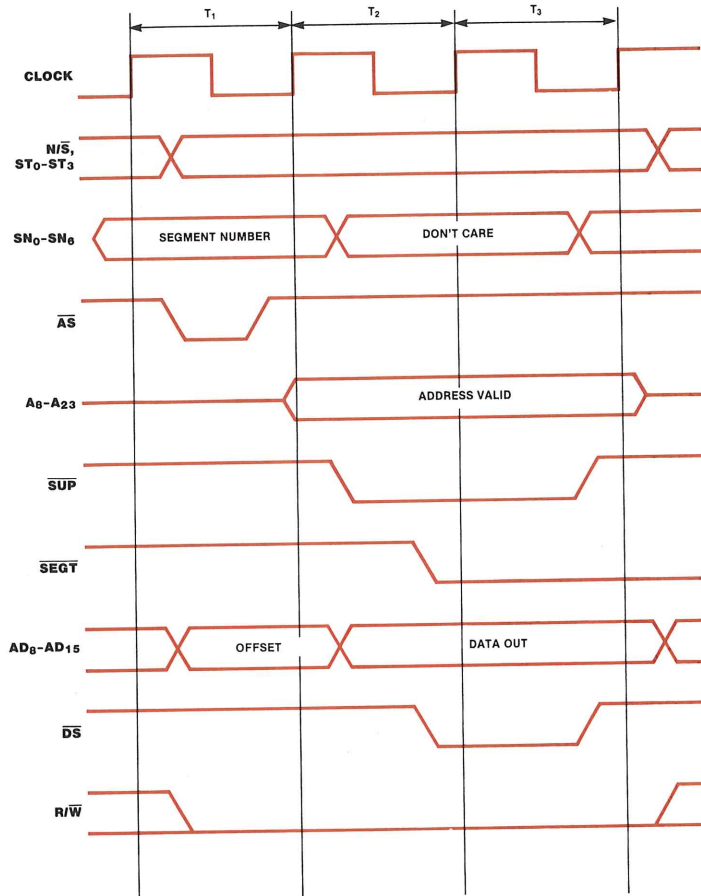


Figure 10. Memory Write Timing

MMU Timing (Continued)

MMU Command Cycle. During the command cycle of the MMU (Figure 11), commands are placed on the Address/Data lines during T_1 . The Status lines indicate that a special I/O instruction is in progress, and the Chip Select line enables the appropriate MMU for that command. Data to be written to a register in the MMU must be valid on the Address/Data lines late in T_2 . Data read from the MMU is

placed on the Address/Data lines late in the T_{WA} cycle.

Input/Output and Refresh. Input/Output and Refresh operations are indicated by the status lines ST_0 – ST_3 . During these operations, the MMU refrains from any address translation or protection checking. The address lines A_8 – A_{23} remain 3-stated.

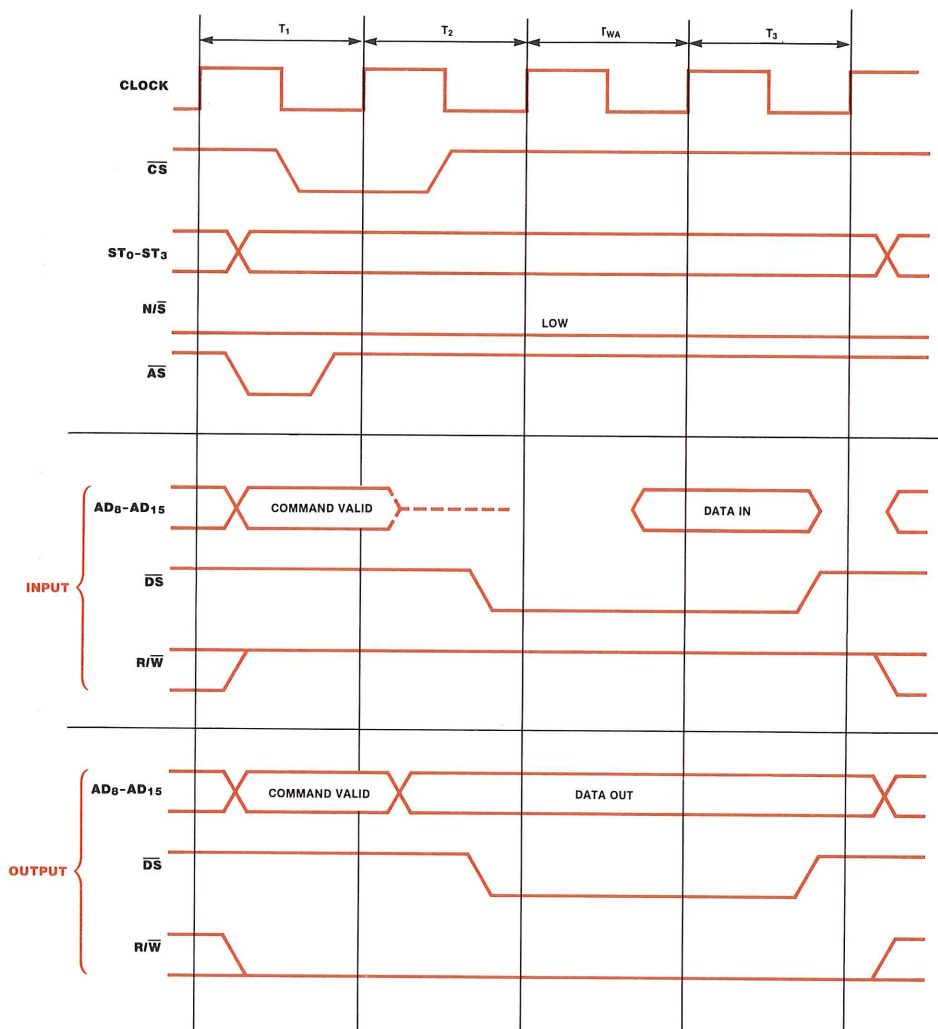


Figure 11. I/O Command Timing

Reset. The MMU can be reset by either hardware or software mechanisms. A hardware reset occurs on the falling edge of the Reset signal; a software reset is performed by a Z8001 special I/O command. A hardware reset clears the Mode Register, Violation Type Register and Descriptor Selection Counter. If the Chip Select line is Low, the Master Enable Flag in the Mode Register is set to 1. All other registers are undefined. After reset, the AD₈-AD₁₅ and A₈-A₂₃ lines are 3-stated. The SUP and SEGT open-drain outputs are not driven. If the Master Enable flag is not set during reset, the MMU does not respond to subsequent addresses on its A/D lines. To enable an MMU after a hardware reset, an MMU command must be used in conjunction with the Chip Select line.

A software reset occurs when the Reset Violation Type Register command is issued. This command clears the Violation Type Register and returns the MMU to its initial state (as if no violations or warnings had occurred). Note that the hardware and software resets have different effects.

Segment Trap and Acknowledge. The Z8010 MMU generates a segment trap whenever it detects an access violation or a write into the lowest block of a segment with the DIRW flag set. In the case of an access violation, the

MMU also activates Suppress. This Suppress signal can be used to inhibit memory writes and to flag special data to be returned on a read access. The Segment Trap remains Low until a Trap Acknowledge signal is received. If a violation occurs, Suppress is asserted for that cycle and all subsequent CPU cycles until the end of the instruction; intervening DMA cycles are not suppressed, however, unless they generate a violation. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only, but no Trap Request is generated.

When the MMU issues a Segment Trap Request it awaits a Segment Trap Acknowledge. Subsequent violations occurring before the Trap Acknowledge is received are still detected and handled appropriately. During the Segment Trap Acknowledge cycle, the MMU drives one of its Address/Data lines High; the particular line selected is a function of the identification field of the mode register. After the Segment Trap has been acknowledged by the Z8001 CPU, the Violation Status Register should be read via the Special I/O commands in order to determine the cause of the trap. The Trap Type Register should also be reset so that subsequent traps will be recorded correctly.

2

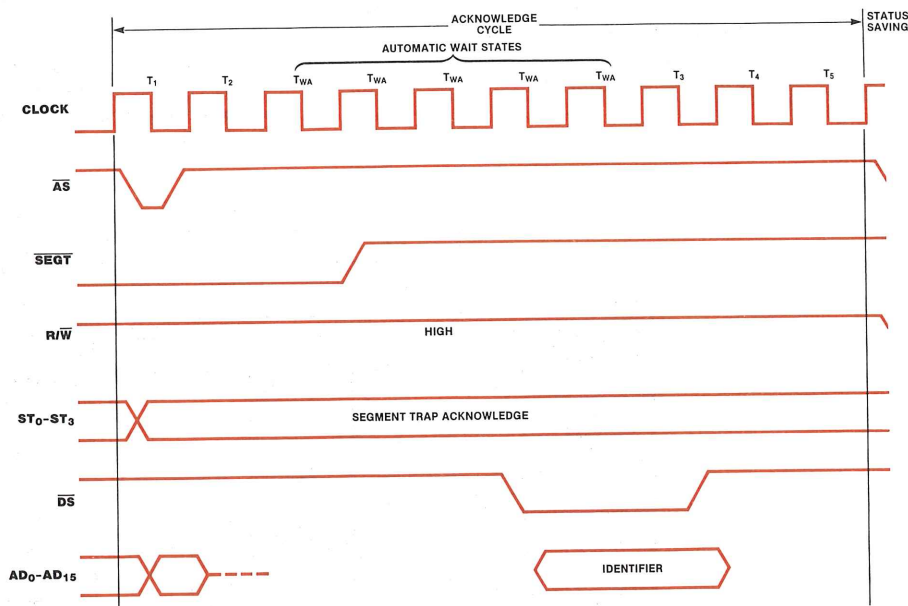


Figure 12. Segment Trap and Acknowledge Timing

**Pin
Description**

A₈-A₂₃. *Address Bus* (outputs, active High, 3-state). These address lines are the 16 most-significant bits of the physical memory location.

AD₈-AD₁₅. *Address/Data Bus* (inputs/outputs, active High, 3-state). These multiplexed address and data lines are used both for commands and for logical addresses intended for translation.

AS. *Address Strobe* (input, active Low). The rising edge of $\overline{AS}$ indicates that AD₀-AD₁₅, ST₀-ST₃, R/W and N/S are valid.

CLK. *System Clock* (input). CLK is the 5 V single-phase time-base input used for both the CPU and MMU.

CS. *Chip Select* (input, active Low). This line selects an MMU for a control command.

Decouple. Output from on-chip negative substrate-bias generator.

DMASync. *DMA/Segment Number Synchronization Strobe* (input, active High). A Low on this line indicates a DMA access is occurring; a High indicates the segment number is valid. It must always be High during CPU cycles.

DS. *Data Strobe* (input, active Low). This line provides timing for the data transfer between the MMU and the Z8001 CPU.

N/S. *Normal/System Mode* (input, Low = System Mode). N/S indicates the Z8001 CPU or Z8016 DMA is in the Normal or System Mode. The signal can also be used to switch between MMUs during different phases of an instruction.

RESET. *Reset* (input, active Low). A Low on this line resets the MMU.

R/W. *Read/Write* (input, Low = write). $R/\overline{W}$ indicates the Z8001 CPU or Z8016 DMA is reading from or writing to memory or the MMU.

SEGT. *Segment Trap Request* (output, active Low, open drain). The MMU interrupts the Z8001 CPU with a Low on this line when the MMU detects an access violation or write warning.

SN₀-SN₆. *Segment Number* (inputs, active High). The SN₀-SN₅ lines are used to address one of 64 segments in the MMU; SN₆ is used to selectively enable the MMU.

ST₀-ST₃. *Status* (inputs, active High). These lines specify the Z8001 CPU status.

ST ₃ -ST ₀	Definition
0 0 0 0	Internal operation
0 0 0 1	Memory refresh
0 0 1 0	I/O reference
0 0 1 1	Special I/O reference (e.g., to an MMU)
0 1 0 0	Segment trap acknowledge
0 1 0 1	Non-maskable interrupt acknowledge
0 1 1 0	Non-vectored interrupt acknowledge
0 1 1 1	Vectored interrupt acknowledge
1 0 0 0	Data memory request
1 0 0 1	Stack memory request
1 0 1 0	Data memory request (EPU)
1 0 1 1	Status memory request (EPU)
1 1 0 0	Instruction space access
1 1 0 1	Instruction fetch, first word
1 1 1 0	Extension processor transfer
1 1 1 1	Reserved

SUP. *Suppress* (output, active Low, open drain). This signal is asserted during the current bus cycle when any access violation except write warning occurs.

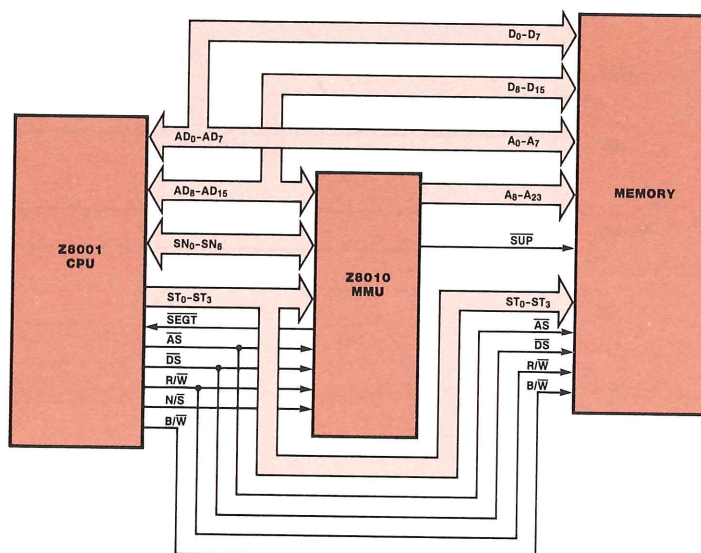


Figure 13. The MMU in a Z8001 System

**Absolute
Maximum
Ratings**

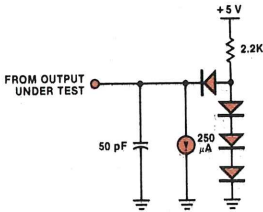
Voltages on all inputs and outputs
with respect to GND.....-0.3 V to +7.0 V
Operating Ambient
Temperature.....0°C to + 70°C
Storage Temperature.....-65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**Standard
Test
Conditions**

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\text{ V} \leq V_{CC} \leq +5.25\text{ V}$
- $\text{GND} = 0\text{ V}$
- $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$



**DC
Character-
istics**

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	$V_{CC}-0.4$	$V_{CC}+0.3$	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\text{ }\mu\text{A}$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0\text{ mA}$
I_{IL}	Input Leakage		± 10	μA	$0.4 \leq V_{IN} \leq +2.4\text{ V}$
I_{OL}	Output Leakage		± 10	μA	$0.4 \leq V_{IN} \leq +2.4\text{ V}$
I_{CC}	V_{CC} Supply Current		300	mA	

AC Character-istics	Number	Symbol	Parameters	Min [3]	Max [3]	Notes
	1	T _c C	Clock Cycle Time	250		
	2	TwCh	Clock Width (High)	105		
	3	TwCl	Clock Width (Low)	105		
	4	TfC	Clock Fall Time		25	
	5	TrC	Clock Rise Time		25	
	6	TdDSA(RDv)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		100	[1]
	7	TdDSA(RDf)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Float Delay	20	75	[1]
	8	TdDSR(RDv)	$\overline{DS}$ ↓ (Read) to AD Output Driven Delay		100	[1]
	9	TdDSR(RDf)	$\overline{DS}$ ↓ (Read) to Read Data Float Delay	20	75	[1]
	10	TdC(WDv)	CLK ↑ to Write Data Valid Delay		160	
	11	ThC(WDn)	CLK ↓ to Write Data Not Valid Hold Time	30		
	12	TwAS	Address Strobe Width	60		
	13	TsOFF(AS)	Offset Valid to $\overline{AS}$ ↓ Setup Time	45		
	14	ThAS(OFFn)	$\overline{AS}$ ↓ to Offset Not Valid Hold Time	60		
	15	TdAS(C)	$\overline{AS}$ ↓ to CLK ↑ Delay	110		
	16	TdDS(AS)	$\overline{DS}$ ↓ to $\overline{AS}$ ↓ Delay	50		
	17	TdAS(DS)	$\overline{AS}$ ↓ to $\overline{DS}$ ↓ Delay	50		
	18	TsSN(C)	SN Data Valid to CLK ↑ Setup Time	120		
	19	ThC(SNn)	CLK ↓ to SN Data Not Valid Hold Time	0		
	20	TdDMAS(C)	DMASync Valid to CLK ↑ Delay	120		
	21	TdSTNR(AS)	Status (ST ₀ –ST ₃ , N/ $\overline{S}$, R/ $\overline{W}$) Valid to $\overline{AS}$ ↓ Delay	60		
	22	TdC(DMA)	Clk ↑ to DMASync ↓ Delay	20		
	23	TdST(C)	Status (ST ₀ –ST ₃) Valid to CLK ↑ Delay	140		
	24	TdDS(STn)	$\overline{DS}$ ↓ to Status Not Valid Delay	0		
	25	TdOFF(Av)	Offset Valid to Address Output Valid Delay		175	[1,4,5]
	26	TdST(Ad)	Status Valid to Address Output Driven Delay		155	[1,4,6]
	27	TdDS(Af)	$\overline{DS}$ ↓ to Address Output Float Delay	30	160	[1]
	28	TdAS(Ad)	$\overline{AS}$ ↓ to Address Output Driven Delay		145	[1, 4]
	29	TdC(Av)	CLK ↑ to Address Output Valid Delay		255	[1, 4]
	30	TdAS(SEGT)	$\overline{AS}$ ↓ to $\overline{SEGT}$ ↓ Delay		160	[1, 2]
	31	TdC(SEGT)	CLK ↑ to $\overline{SEGT}$ ↓ Delay		300	[1, 2]
	32	TdAS(SUP)	$\overline{AS}$ ↓ to SUP ↓ Delay		150	[1, 2]
	33	TdDS(SUP)	$\overline{DS}$ ↓ to SUP ↓ Delay	30	155	[1, 2]
	34	TsCS(AS)	Chip Select Input Valid to $\overline{AS}$ ↓ Setup Time	10		
	35	ThAS(CSn)	$\overline{AS}$ ↓ to Chip Select Input Not Valid Hold Time	60		
	36	TdAS(C)	$\overline{AS}$ ↓ to CLK ↑ Delay	0		
	37	TsCS(RST)	Chip Select Input Valid to $\overline{RESET}$ ↓ Setup Time	150		
	38	ThRST(CSn)	$\overline{RESET}$ ↓ to Chip Select Input Not Valid Hold Time	0		
	39	TwRST	$\overline{RESET}$ Width (Low)	2T _c C		
	40	TdC(RDv)	CLK ↑ to Read Data Valid Delay		460	[1]
	41	TdDS(C)	$\overline{DS}$ ↓ to CLK ↑ Delay	30		
	42	TdC(DS)	CLK ↓ to $\overline{DS}$ ↓ Delay	0	110	

[1] 50 pF Load.

[2] 2.2K Pull-up.

[3] All values in nanoseconds.

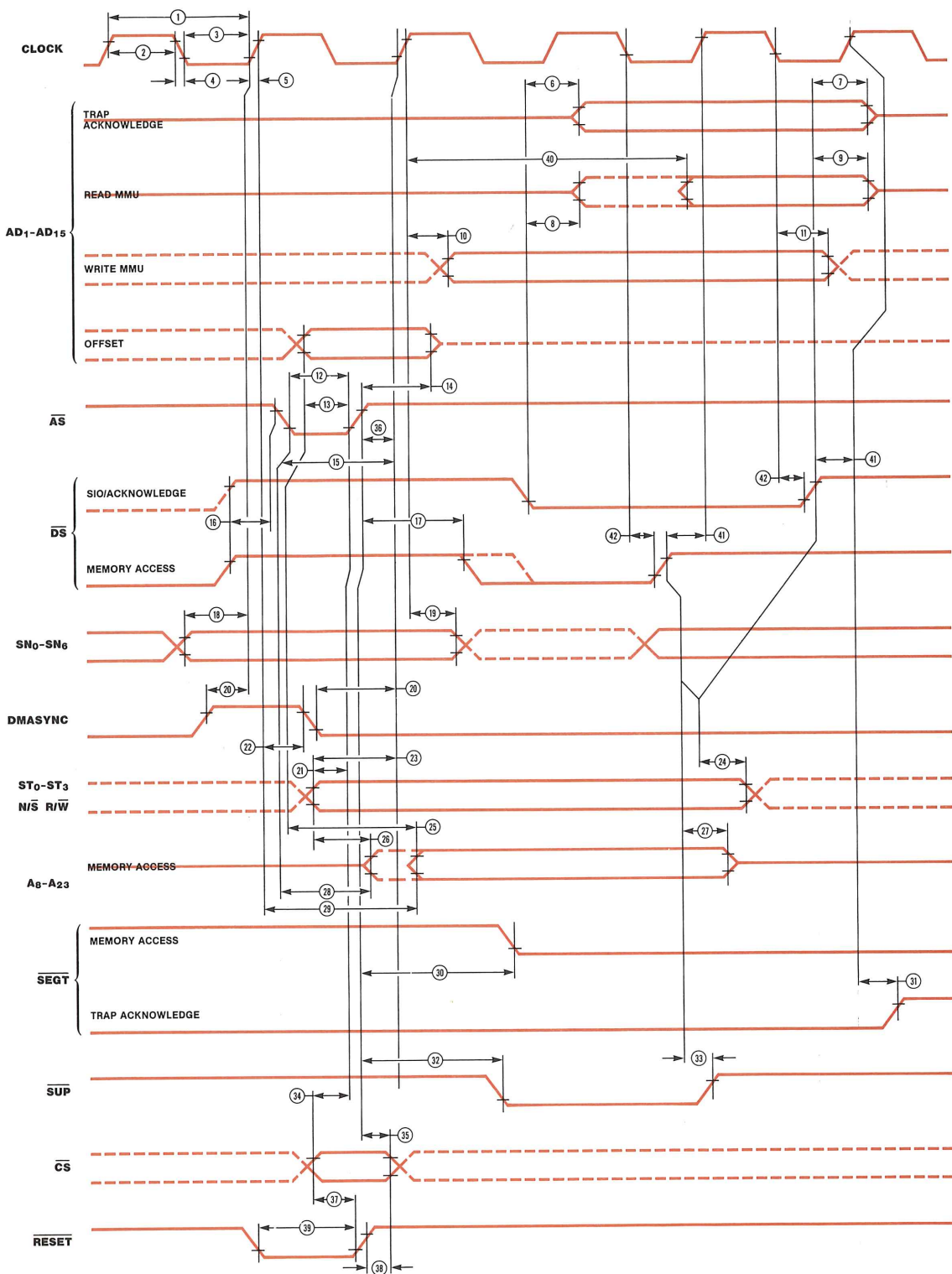
[4] These values apply to the Z8010-3 version only. A Z8010-2 version available soon will improve these values by 40 ns.

[5] Parameter 9 (Clock ↑ to Address Valid) in the Z8001/2 CPU

Product Specification is specified at 100 ns with 100 pF load.

With a load of 50 pF, this delay is reduced to 80 ns.

[6] Parameter 47 (Clock ↑ to Status Valid Delay) in the Z8001/2 CPU Product Specification is specified at 110 ns with a 100 pF load. With a 50 pF load, this delay is reduced to 100 ns.



2

An Introduction to the Z8010 MMU Memory Management Unit



NEW
1980

Tutorial Information

February 1980

2

Introduction

The declining cost of memory, coupled with the increasing power of microprocessors, has accelerated the trend in microcomputer systems to the use of high-level languages, sophisticated operating systems, complex programs and large data bases. The Z8001 microprocessor supports these advances by offering multiple 8M byte address spaces as well as a rich and powerful instruction set. The Z8010 Memory Management Unit (MMU) supports the Z8001 processor in the efficient and flexible use of its large address space.

Support for managing a large memory can take many forms:

- Providing a logical structure to the memory space that is largely independent of the actual physical location of the data
- Protecting the user from inadvertent mistakes such as attempting to execute data

Motivations for Memory Management

The primary memory of a computer is one of its major resources. As such, the management of this resource becomes a major concern as demands on it increase. These demands can arise from different sources, three of which are of interest in the present context. The first stems from multiple users (or multiple tasks within a dedicated application) contending for a limited amount of physical memory. The second comes from the desire to increase the integrity of the system by limiting access to various portions of the memory. The final source arises from issues surrounding the development of large, complex programs or systems. Each of these three sources involves a multifaceted group of related issues.

When multiple tasks constitute a given system (for example, multiple users of a system or multiple sub-tasks of a dedicated application), the possibility exists that not all tasks may be in primary memory at the same time. (A task is the action of executing a program on its data; a task may be as simple as a single

- Preventing one user from unauthorized access to memory resources or data
- Protecting the operating system from unexpected access by the users.

The Z8010 provides all these features plus additional features that permit a variety of system hardware configurations and system designs.

This paper examines the various uses of memory management in computer systems and how memory management techniques generally meet these requirements. The major features of the Z8010 MMU illustrate how memory management functions can be supported by hardware. A few examples demonstrate how this LSI circuit can be used to configure several different memory management systems.

procedure or as complex as a set of related routines.) If the population of memory-resident tasks can vary over time, a useful feature of a system would be the ability for a task to reside anywhere in memory, and perhaps in several different locations during its lifetime. Such tasks are called *relocatable*, and a system in which all tasks are relocatable generally offers greater flexibility in responding to changing system environments than a system in which each task must reside in a fixed location.

A second issue that arises in multi-task environments is that of sharing. Separate tasks may execute the same program on different data, and may therefore share common code. For example, several users compiling FORTRAN programs may wish to share the compiler rather than each user having a separate copy in memory. Alternatively, several tasks may wish to execute different programs using the same data as input, and it may be possible for these tasks to access the same copy of the

input. For example, a user may wish to print a PASCAL program while it is being compiled; the print process and the compiler process could access the same copy of the text file.

A third issue in multi-task systems is protecting one task from unwanted interactions with another. The classic example of unwanted interaction is one user's unauthorized reading of another user's data. Prohibiting all such interactions conflicts with the goal of sharing and so this issue is usually one of selectively prohibiting certain types of interactions. The issue of protecting memory resources from unauthorized access is usually included in the larger set of issues relating to system integrity.

System integrity takes many forms in addition to protecting a task's data from unwanted access. Another aspect is preventing user tasks from performing operating system functions and thereby interrupting the orderly dispatch of these tasks. For example, most large systems prevent a user task from directly initiating I/O operations because this can disrupt the correct functioning of the system.

Another aspect of separating users from system functions relates to separating system I/O transfers from user tasks, especially with respect to error conditions. For example, an error during a direct memory access, say to a nonexistent memory location, should not cause an error in the program that is currently executing.

A final example of increasing the system integrity is protecting a user task from itself. Obvious errors, such as trying to execute data or overflowing an area set aside for a stack, can be detected while a program is executing and handled appropriately, provided the system is given sufficient information.

The notion of protecting an executing task from performing certain types of actions known to be erroneous introduces a third general motivation for memory management, namely support for the design and correct implementation of large, complex programs and systems.

Protecting a task from itself obviously helps in debugging a large program, but there are other system features that can aid in developing complex systems. Modern methodology for developing large systems dictates partitioning a task into a number of small, simple, self-contained sub-tasks with well defined interfaces. Each sub-task generally interacts with only a few other sub-tasks and this communication is carefully controlled. This methodology promotes a systems design that can be readily modified, but it also tends to promote the creation of a large number of nearly independent sub-tasks and many data structures accessible to only one or a few of these sub-tasks. Because modern systems are increasingly driven to support many interacting tasks, possibly written and compiled separately, they must also enforce some communication protocol without sacrificing efficient operation. Modern memory management systems can offer effective tools for implementing large systems designed using this methodology.

In summary, the major goals of memory management systems are to:

- Provide flexible and efficient allocation of memory resources during the execution of tasks
- Support multiple, independent tasks that can share access to common resources
- Provide protection from unauthorized or unintentional access to data or other memory resources
- Detect obviously incorrect use of memory by an executing task
- Separate users from system functions.

Most of today's memory management systems support these functions to some degree. The extent of this support is largely a question of resources to be devoted to these functions and the understood demands of the intended applications for these systems.

**The Fundamentals of
Memory
Management**

Memory management has two functions: the *allocation* and the *protection* of memory. Dynamic relocation of tasks during their execution is accomplished by an address translation mechanism. The restriction of memory access is accomplished by memory attribute checking. Both operations occur with each memory request during the execution of a program and both are transparent to the user.

Address translation simply means treating the memory addresses generated by the program as logical addresses to be interpreted or translated into actual physical memory locations before dispatching the memory access requests to the memory unit. Memory attribute checking means that each area of memory has associated with it information as to who can

access it and what types of access can be made by each task. Each memory reference is checked to insure that the task has the right to access that location in the given fashion (for example, to read the contents of the location or to write data to that location).

Instead of a linear address space, more elaborate memory management systems have a hierarchical structure in which the memory consists of a collection of memory areas, called segments. Access to this structured memory requires the specification of a segment and an offset within that segment. Thus, instead of specifying memory location 1050 in a linear address space, a task specifies memory location 5 in segment number 23, for example.

Generally, segments can be of variable size, within limits, and a user can specify the size of each segment to be used. Thus one user may have two segments of two thousand and ten thousand words for his FORTRAN program and data, respectively, while another user might have three segments of three thousand, six thousand and two thousand words for her PASCAL program, data, and run-time stack. If the first user called his data segment number 5, then the first word in his data set would be accessed by the logical address (5,0) indicating segment 5, offset 0. The memory management system translates this symbolic name into the correct physical memory address.

Figure 1 gives a conceptual realization of these two users' logical program spaces. The first user, User A, has his program segment called "Segment 6" and his data segment called "Segment 5." The second user, User B, has her program segment called "Segment 5," her data segment called "Segment 12" and her stack segment called "Segment 2." Notice that both users have named one of their segments "Segment 5," but they refer to different entities. This causes no problem since the system keeps the two memory areas separate. The situation is analogous to both users having an integer variable called "I" in their programs: The system realizes that these are two separate variables stored in different memory locations.

User A's data segment, "Segment 5," is ten thousand words. If he references word 10,050

of Segment 5 he gets an error message from the system indicating that he has exceeded the allocation limit for Segment 5. Note that he does not access word 50 of Segment 6. That is, segments are logically distinct and unordered. A reference to one segment cannot inadvertently result in access to another segment. Thus, in this example, User A is prevented from accidentally (or deliberately) accessing his program as though it were part of his data segment.

Figure 2 illustrates one way that these segments could be arranged in the physical memory. The dotted lines indicate the memory-mapping function from the logical address space of the user to the physical memory locations allocated to him. The figure also indicates the access attributes associated with each user's segments. For example, program segments are "execute only" and data segments are "read/write." Thus a user is prevented from executing a data segment or writing into a code segment.

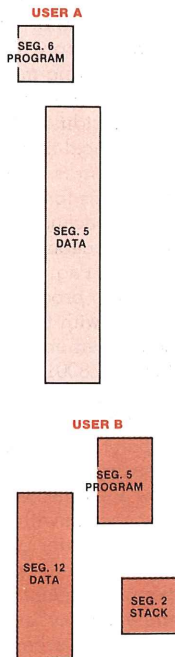


Figure 1. Two User's Logical Address Space

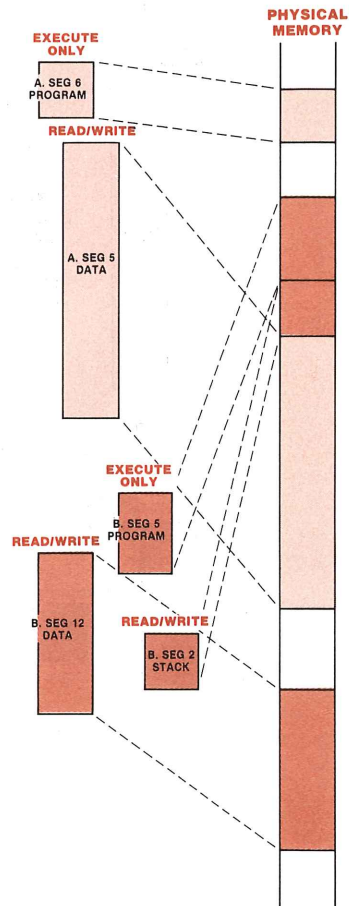


Figure 2. Mapping Logical Segments to Physical Memory

The Fundamentals of Memory Management

(Continued)

Figure 3 illustrates what happens when both users have access to the same data set in primary memory, say the results of a questionnaire that both intend to analyze. Each user has a logical name associated with that data set to specify the segment in which the data set is to reside. Note that the two users have chosen to put the data set in different segments of their personal address spaces. The system-mapping function translates these different segment names to the same physical memory locations. Thus User A's access to address (2, 17) references the same physical memory location as User B's access to address (7, 17). In the figure, note that two of B's segments have been moved in physical memory to create a space large enough to hold the questionnaire data.

Another topic in memory management that is supported by Z8001-Z8010 architecture but requires additional support hardware is demand swapping, or segmented virtual memory, which means that the logical memory

area may not actually reside in physical memory until a task actually tries to access it. At the time an access is made to a segment missing from physical memory, the instruction execution is held in abeyance until the logical memory can be brought into the physical memory and then the instruction is allowed to proceed with the memory access. The address translation is performed, access protection is checked and the instruction proceeds as if the logical memory area had been in the physical memory at the beginning of the instruction. The instructions in the Z8001 must run to completion before the CPU can perform any action, such as responding to a missing segment trap. But with the conjunction of hardware and software to simulate the above functions, a segmented virtual memory scheme can be implemented.

A final topic in memory management is paging, which is another method for partitioning a user address space and mapping it onto the physical memory. Paging is most effective when demand swapping can be supported. Essentially, paging divides the logical memory into fixed-size blocks, called pages. Like segments, the individual pages can be located anywhere in the physical memory and a translation mechanism maps logical addresses to physical memory locations. There are two differences between paging and segmenting a logical memory. First, pages are of fixed size whereas segments are of various sizes. Second, under paging, the logical memory is still linear, that is, a task accesses memory using a single number, rather than a pair as in segmentation. The major advantage of paging is in treating memory as blocks of fixed sizes, which simplifies allocating memory to users and deciding where to place the logical pages in physical memory. The major disadvantage of paging is in assigning different protection attributes to different areas in a user address space because a paged memory appears homogeneous to the user and the operating system. Paging can be combined with segmentation to produce a memory management system with the advantages of both paging and segmentation. The implementation of paging for the Z8001 requires additional support hardware and may be implemented independent of the Z8010.

Before proceeding to the mechanism of memory management, it is instructive to review how a segmented address translation mechanism with protection attributes achieves the five major goals of memory management outlined in the previous section. The first goal permits dynamic allocation of memory during the execution of tasks; that is, a task could be located anywhere in memory and even moved about when its execution is suspended. The address translation mechanism provides this flexibility because the task deals exclusively

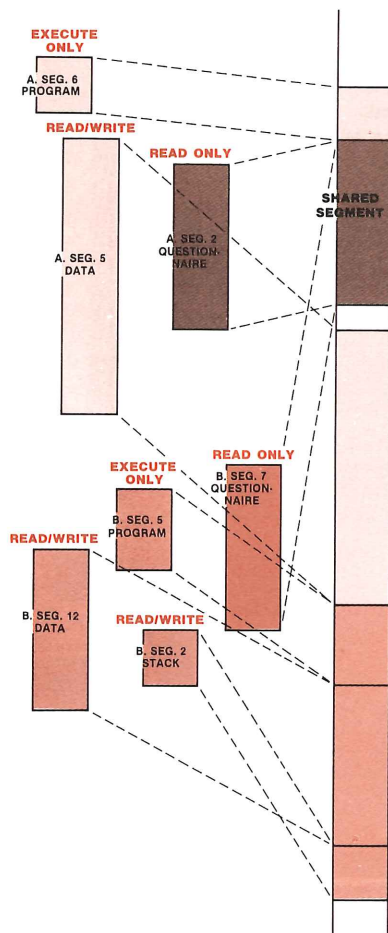


Figure 3. Two Users Sharing a Common Segment

The Fundamentals of Memory Management

(Continued)

with logical addresses and hence is independent of the addresses of the physical memory locations it accesses. Moving the task to different physical memory locations requires that the address mapping function be changed to reflect the change in memory location, but the task's code need not be modified. Of course, this flexibility does incur the price of managing the various system tables required to implement memory management.

The second goal supports sharing of common memory areas by different tasks. This is accomplished by mapping different logical areas in different tasks to the same physical memory locations.

The third provides protection against certain types of memory accesses. This is accomplished by associating accessing attributes with each logical segment and checking the type of access to see if each access is permitted.

The fourth goal detects obvious execution errors related to memory accessing. This can be accomplished by checking each access to a segment to see whether the address falls within the allocated physical memory for that segment. It could also include affixing a read/write attribute to data to prevent a task from trying to execute a data segment, and affixing an execute-only attribute to code segments to prevent a task from trying to read or write data to this segment. Additionally, if a segment is used for a stack, the system could issue a warning to a task when the stack approaches the allocated limit of the segment. The task could then request more memory for the stack before the stack overflows and creates a fatal error.

The final goal listed for memory manage-

ment systems separates user functions from system functions. For processors that distinguish between System mode and User mode of operation, this goal can be accomplished by associating a system-only attribute with system segments so users cannot directly access system tables and tasks.

As a final point, it should be noted how segmentation can be used to support the development and execution of large, complex programs and systems. The concept of segmentation corresponds to the concept of partitioning a large system into procedures and data structures where each procedure and data structure can be associated with a separate segment. A task can then invoke a procedure or sub-task or access a data structure by referring to its logical segment name. Access to these objects can be individually restricted by using the protection-checking mechanism of the memory management system.

As a specific example of how segmentation could be used in the design of a large system, consider a multi-user interactive BASIC system with a large data base shared by all users. Such a system could be designed with segments 0 through 15 reserved for system use, segments 16 through 31 reserved for the BASIC interpreter and its internal tables, segments 32 through 63 allocated to user tasks and segments 64 through 127 reserved for portions of the data base when they are in primary memory being accessed by users. For this system, segments 0 through 31 would probably always be in memory; the other segments would be assigned as needed and the memory they require allocated dynamically.

2

The Mechanics of Memory Management

Essentially there are four issues in implementing a memory management system: how addresses are specified, how these addresses are translated, what attributes are checked for each access, and how the protection mechanism is implemented. Some of the major alternatives in each of these issues are briefly discussed here, primarily from the point of view of a segmented memory.

Two approaches have traditionally been taken for specifying addresses in a segmented memory. For simplicity, only addresses in instructions are discussed. The first way puts all the addressing information in the instruction itself. That is, each memory address in an instruction contains both the segment name and the offset within the segment. The alternative sets aside special registers that contain some of this information, for example the segment name or the address in physical memory where the segment resides.

The advantage of the latter approach lies in the fact that fewer bits are needed in an instruction to specify addresses. Thus programs may be shorter. Also, because there is

reduced traffic between the memory and the processor for fetching shorter instructions, a program may execute faster.

On the other hand, these special registers must be manipulated to access more segments than there are registers, and this manipulation adds to the number of instructions, the program size and the execution time. In practice, these can destroy the advantages described above. If the special registers contain physical memory locations, then these must be protected from user access to maintain the integrity of the system, and changing segments requires system calls which can be time consuming if too few registers are supplied. The Z8001 architecture specifies the complete logical address in the instruction.

Address translation is performed by adding the logical segment offset to the memory location where the segment begins. Thus, when an address of the form (a, b) is presented to the translation mechanism, the segment name "a" is used to determine where segment "a" resides in memory. Assume that it resides in locations 10000 to 25000. Then the actual

The Mechanics of Memory Management (Continued)

memory location of (a, b) is memory location $10000 + b$. The major option in implementing this type of address translation is in determining the segment location in physical memory. When special registers have been set aside to contain the starting location of the segment instead of putting all address information in the instruction, the addressing mechanism is similar to using the segment register as an index register or a base register.

When logical addresses are either completely specified in the instruction or when the special register contains the symbolic segment name, a table must be used to translate the logical segment name into a physical memory location. The table may have an associative capability, that is, the segment name is presented to the table and the device returns the physical memory location where the segment begins. Alternatively, the table could have one entry for every possible segment name. The Z8010 implementation of the address translation table sets aside a specific table entry for each logical segment name.

A number of attributes can be associated with a segment and checked during each access. One of these is the allocated length of the segment, and each access is checked to see if it falls within the bounds of the segment. The Z8010 provides limit checking.

Another type of attribute deals with ownership or class of ownership: tasks are grouped into classes and only those in certain classes are permitted access. The simplest example is the system versus user classification, where tasks are either one or the other and this determines whether or not any type of access can be made to the segment. The Z8010 has this feature—users are prevented from accessing system segments.

Other types of attributes that can be associated with a segment involve modes of accessing, for example read only, read/write or execute only. For these attributes, the processor must indicate the type of access to be made, be it code fetch, read from memory, write to memory, etc. The Z8001 indicates when it is fetching code, reading or writing data, or performing stack operations, and thus the Z8010 can offer protection for these opera-

tions. The other issue with respect to attributes is whether they are permissive or prohibitive. That is, whether the attribute is in the form of "write to this segment is permitted" or of the form "write to this segment is prohibited." The Z8010 adopts the approach of specifying attributes that prohibit certain types of accessing.

The final issue in the mechanics of memory management systems is the implementation of the protection attributes. These may be associated either with the logical address space or with the physical memory itself. The IBM 360 series, for example, places the memory protection information with the physical memory itself. Thus the processor generates a memory address and the memory module checks to see if the access is permitted. The main difficulty with this approach is in the lack of flexibility, because protection is associated with fixed memory partitions. Also, sharing memory is cumbersome because each user is given a protection key to match the memory key; thus both users must have the same access key or a universal access key. Associating access attributes with the logical segment permits a versatile memory management scheme because different users can access the same segment and have different access attributes associated with their accessing. The Z8010 implements access attributes using the segment mapping information.

Other information associated with each segment does not pertain to the protection mechanism but can be of use to the memory management system. This information generally relates to the history of the segment; for example, whether a segment has been modified while resident in primary memory. If it has not been modified and the system requires the memory for another segment, the memory can be freed immediately; otherwise, the updated version of the segment must be stored in secondary memory and the primary memory is not available until the segment has been saved. Although not strictly necessary, such information can improve the performance of the memory management system. The Z8010 collects information on segment usage, and this information can be used to enhance performance of systems that use this device.

The Z8010 Memory Management Unit

The Z8001 CPU generates segmented addresses consisting of a 7-bit segment number and a 16-bit segment offset address. In addition, the CPU generates status signals indicating its current mode of operation (such as Instruction Fetch, Data Memory Reference, Stack Memory Reference, and Internal Operation), whether it is performing a Read or a Write Memory Reference and whether it is in Normal (User) or System Mode. The Z8010 Memory Management Unit uses this information to perform its memory management functions. This section describes the Z8010 MMU in

some detail, beginning with the translation procedure and continuing with a description of the internal registers of the chip. The section concludes with a description of the system commands that alter the contents of these registers.

The Z8010 MMU has three functional states. The first is the memory management state: when a logical address is presented to the unit, the MMU checks the access to insure its validity and translates the logical address to a physical memory location. The second state is a command state: when a special I/O instruc-

tion is issued to the MMU, such as reading or writing one of its internal registers, the MMU responds to the command as appropriate. The third state is a quiescent state: when the CPU issues an I/O instruction or a refresh cycle, the MMU address lines remain 3-stated.

The inputs to the MMU are the Address/Data lines (A/D lines), Segment Number lines, Bus Status and Timing Lines, and special control lines for chip selection and DMA. The outputs from the MMU are Address lines, a Segment Trap line and a Suppress line (Figure 4). During address translation and access protection, logical addresses are presented to the MMU on the Segment Number and Address/Data lines; the MMU puts the translated physical memory location on its Address lines and, if appropriate, activates the Segment Trap and/or Suppress lines.

Segment Trap is a special type of synchronous interrupt for the Z8001 CPU; Suppress aborts the memory access. In the command state, the MMU receives commands on the A/D lines; data to be read from or written into the MMU is also placed on the A/D lines.

The MMU selects which of the three states it will be in according to the status information on the Bus Status lines during the initial clock cycle of an instruction or DMA cycle. The MMU performs address translation during a memory reference for either a regular instruction or a DMA request. Only I/O instructions (either regular or special), memory refresh and reserved bus status states cause the MMU to cease performing memory address translations and enter another state.

The MMU uses the segment number to access an internal table of segment descriptor registers, each register containing the starting memory location of the segment (called the base address), the segment's limit (used to determine the range of legal address offsets) and the types of accesses permitted to that segment.

Physical memory for segments is allocated in blocks of 256 bytes. The eight least significant bits of the base address are all zero and are not stored in the Segment Descriptor Register. Also, since the eight low-order bits of the segment base are always zero, the eight low-order bits of the segment offset need not participate

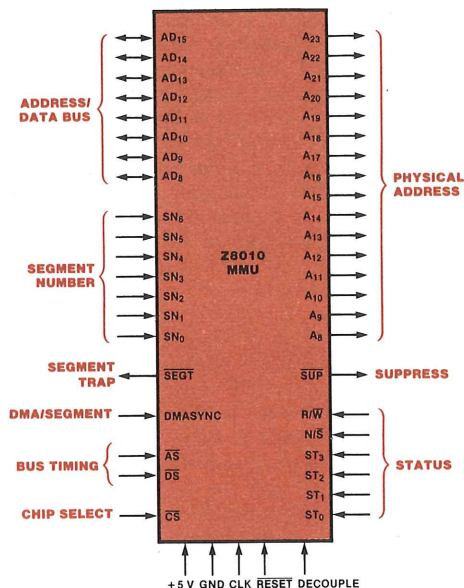


Figure 4. Z8010 MMU Pin Functions

in the addition of the base address to the offset. Rather, they can be juxtaposed to the result of adding the high-order byte of the offset to the most significant 16 bits of the base address.

This process is illustrated in Figure 5. Note that the low-order eight bits of the offset are not used by the MMU. Figure 6 goes through an example of mapping the logical address (5, 1528) to a physical memory location when segment 5 begins at location 231100.

Figure 6a illustrates the full addition to be performed during address translation. The segment number 5 selects Segment Descriptor Register 5 in the MMU. The base address field in this register contains 2311 which corresponds to a base address of 231100. The offset, 1528, is then added to 231100 to produce the physical memory location 232628. Figure 6b represents the same logical procedure, but illustrates the actual operation of the MMU. Again segment number 5 is used to select the base address. However, only the high-order byte of the offset is added to the contents of the

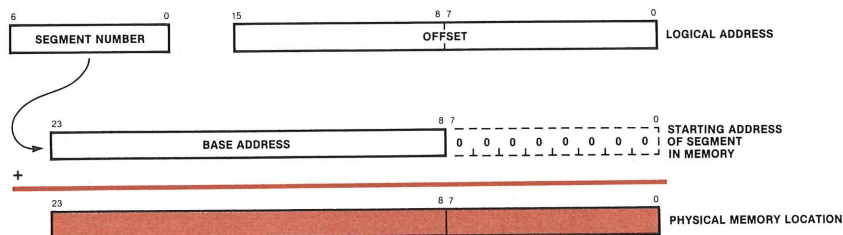


Figure 5. Generation of the Physical Memory Location from a Logical Address

The Z8010 Memory Management Unit (Continued)

MMU base-address field: 15 is added to 2311 to produce the most significant 16 bits of the physical memory location. The low-order byte of the physical location is the same as the low-order byte of the offset.

The results of the two processes illustrated in figures 6a and 6b are the same, but in 6a a 24-bit addition is implied whereas in 6b only a 16-bit addition is needed. Also, the low-order eight bits of the offset are not needed by the MMU and this reduces the number of pins required by the MMU package.

The MMU checks memory references for two types of trap conditions. The first type is an access violation. This occurs when a memory reference is performed in a mode that is not allowed by the read-only, execute-only, CPU-inhibit or system-only attribute of a segment. A memory reference outside the allocated memory for the segment also constitutes an access violation.

The second type is a write warning. This occurs when a write is made to the last 256 bytes of a special type of segment (indicated by a special attribute flag called the Direction And Warning Flag). These segments are typically used for stacks and are therefore logically organized so that successive writes (or stack pushes) access lower-numbered memory locations. By generating a segment trap request when a write is performed into the lowest-numbered 256 bytes of the memory allocated for these segments, the MMU is signaling that a stack is in danger of overflowing. The operating system in servicing this trap can increase the memory allocated for the segment and avoid a fatal stack overflow condition.

The MMU generates two control signals that can be used by the system to perform memory management functions. Segment Trap Request is generated upon the first detected occur-

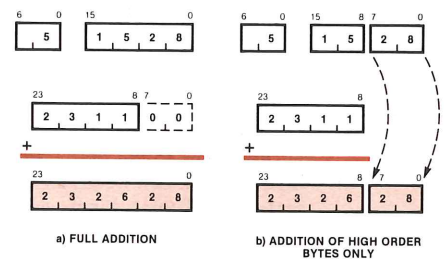


Figure 6. Two Methods of Address Translation

range of a violation or write warning. Once asserted, this signal remains set until a trap acknowledge signal is received. Only when the Fatal Flag, a special MMU control flag, is set will a detected violation not cause a segment trap request. This flag is set only when a second violation is detected while a previous trap is being processed and thus indicates that the system software is in error.

The other control signal generated by the MMU is Suppress. Once a violation has been detected, this signal is asserted on that and every succeeding memory reference for the remainder of the instruction. In particular, I/O and Special I/O instructions are checked for memory access violations, and once a memory access violation is detected, subsequent memory accesses cause Suppress signals to be generated. I/O addresses, of course, bypass the MMU and are neither translated nor checked. Intervening DMA cycles and memory refresh cycles are exceptions to this rule. During such cycles Suppress is not asserted unless a violation is detected during that cycle. Only DMA can generate a violation; refresh can never cause a violation. Suppress can be used by the memory system to inhibit writes, thus protecting the memory from illegal alterations.

MMU Internal Registers

There are three groups of registers in the MMU: Segment Descriptor Registers, Control Registers and Status Registers. The Segment Descriptor Registers contain all the information relating to the address translation and access protection of a particular segment. The Con-

trol Registers contain information used to control the various functions of the MMU, including how to interpret various signals generated by the CPU. The Status Registers contain all the information the MMU generates when it detects an access violation.

Segment Descriptor Registers

Because there are 64 Segment Descriptor Registers in the MMU, two MMUs are required to handle all 128 segments that the Z8001 can manipulate directly. An MMU is programmed to handle either segments 0 through 63 or segments 64 through 127; the particular set of 64 segments in an MMU can be changed using special operating system commands. Each Segment Descriptor contains three fields, a 16-bit Base Field, an 8-bit Limit Field and an 8-bit Attribute Field (Figure 7). The segment number of a logical address determines which

segment descriptors are used in address translation.

The Base Field specifies the starting location in memory of the segment.

The Limit Field specifies the segment size in blocks of 256 bytes. The address offset is compared against the segment limit and a size violation occurs if the offset falls outside the segment boundaries. A write warning occurs if the destination is in the last block of a segment being used as a stack.

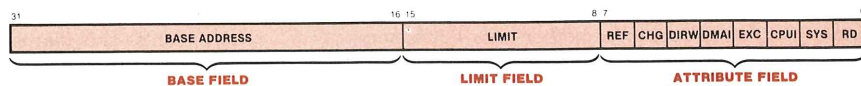


Figure 7. A Segment Descriptor

The *Attribute Field* contains eight flags. Five flags protect the segment against certain types of access, one indicates a special orientation of the segment, and two indicate the types of accesses that have been made to the segment. The following brief description explains how these flags are used.

The *Read-Only Flag (RD)* indicates that the only accesses to this segment are reads. Writes are prohibited when this flag is set. Thus this flag is a write-inhibit flag; in particular, code can be executed from a read-only segment. This flag is useful in protecting data from being written by unauthorized users. For example, if one user wants to give another access to a document that he has created, but does not want this user to be able to modify it, the system can set the Read-Only Flag when it copies the file into the user's address space. If the data is already in memory (in a read-only mode), then this same memory area can be made accessible to that user without another copy of the document being required.

The *System-Only Flag (SYS)* indicates that only accesses made in System Mode are to be permitted. When this flag is set, accesses in the Normal Mode are prohibited. This attribute is useful in protecting system tables and tasks from being accessed by users. For example, system I/O routines can be left in the memory with this flag set and a user is unable to call them directly. This feature is useful if a system is designed so that users are given certain segment names and other segment names are reserved for system use. This flag prevents users from accessing system segments, even though they can generate the logical addresses.

The *CPU-Inhibit Flag (CPUI)* indicates that the segment is not to be referenced by the CPU. When this flag is set, CPU access to this segment is prohibited, but DMA channels can access the segment. This flag is useful in preventing a program from accessing a segment whose data resides on secondary storage and has not been brought into primary memory. For example, a user may request the operating system to read a file from disk into segment number 19; if the operating system returns control to the user before the file has been read, this flag should be set in Segment Descriptor Register 19.

The *Execute-Only Flag (EXC)* indicates that the segment is to be referenced only during the instruction fetch cycle of the processor. When this flag is set, access to the segment during any other cycle of an instruction, for example during the memory request cycle, is

prohibited. This flag is useful in preventing a program from making a copy of a proprietary program. For example, if this flag is set for a segment containing code that a user can access, that code is protected from being read and hence from being copied.

The *DMA-Inhibit Flag (DMAI)* indicates that the segment is not to be referenced by a DMA Channel. When this flag is set, only the CPU has access to the segment. This flag is useful in preventing a DMA device from modifying a segment being used by an executing task. For example, segments with valid data should have this flag set to protect them from modification by a DMA device.

The *Direction And Warning Flag (DIRW)* indicates that memory accesses are to be monitored and certain accesses are to be signaled, although allowed to proceed. When this flag is set, any write to the lowest 256 bytes of the segment generates a write warning. This flag is useful for segments that are used as stacks since the Z8001 has special stack instructions to manipulate stacks that grow toward lower memory locations. Thus a write warning for a stack indicates that the stack may soon overflow its allotted memory space and that more physical memory should be obtained. For example, if a segment serves as a run-time stack for a block-structured programming language such as PASCAL, memory can be allocated to this segment only as a program requires during its execution. The alternative in a fixed allocation environment is to allocate as much memory for the stack as the system expects the program to need, whether or not it is actually used by the program.

The *Changed Flag (CHG)* indicates that a write has occurred to this segment. This flag is set automatically whenever a program or DMA device writes into the segment. This flag is useful in indicating which segments have been modified in the case where the segment must be written to a secondary storage device. Segments that have not been updated need not be copied back to disk if a copy already exists. For example, when a user task is suspended in a multiple-user environment and his task is to be swapped out of memory temporarily to make room for another task, only those segments that have been changed need to be updated on the disk.

The *Referenced Flag (REF)* indicates that a memory access has been made to a segment. This flag is set automatically whenever a program or DMA device accesses the segment. This flag is useful in indicating which segments are active in the case that a segment must be

selected to be swapped out of primary memory to make room for another task. For example, seldom-used operating-system tasks that usually reside in primary memory may be swapped

out to make room for users with large memory requirements. This flag is a way of ascertaining which segments contain seldom used tasks.

**Control
Registers**

Three user-accessible 8-bit registers in the MMU control the functioning of the MMU (Figure 8). The Mode Register provides a sophisticated method for selectively enabling MMUs in a multiple-MMU configuration. The Segment Address Register (SAR) selects a particular segment descriptor to be accessed by a system routine when it is changing the organization of primary memory. The Descriptor Selection Counter Register selects the particular byte in the Segment Descriptor Register that is accessed.

Two flags in the Mode Register govern the functioning of the MMU. The Master Enable Flag (MSEN) indicates whether the device will perform address translation. When this flag is set, addresses translated by the MMU are placed on its Address lines; when this flag is clear, the Address lines are 3-stated. Thus, once this flag is reset, no memory request can pass through the MMU. In a single-MMU configuration, MSEN set to zero requires that the CPU must have access to a special memory, since it will not be able to fetch an instruction from the primary memory. This flag can be set during hardware reset (this is discussed later).

The second flag in the mode register that governs the functioning of the MMU is the Translate Flag (TRNS). This flag indicates whether the MMU is to translate the addresses presented to it. When the flag is set, the MMU translates logical addresses to physical memory locations and checks to see if a violation will occur on that access. When the flag is clear, addresses presented to the MMU are passed to the output Address lines without change, and no protection checking is done.

When multiple-MMUs are used in a memory-management system, some mechanism must be present to select those devices that are to be active during the memory translation process. More specifically, if two MMUs are employed so that all 128 segments can be used at random by an executing process, then some way must exist for each of the MMUs to know which 64 Segment Descriptors are located in its Segment Descriptor Registers. The Upper Range Select Flag (URS) indicates which set of 64 descriptors is stored in the MMU. When the flag is set, the MMU contains descriptors 64 through

127; when the flag is reset, the MMU contains descriptors 0 through 63.

When multiple-MMU devices keep separate tables for system descriptors and user descriptors, the Multiple Segment Table Flag (MST) and the Normal Mode Select Flag (NMS) in the Mode Register distinguish which MMUs contain system descriptors and which contain user descriptors. When the MST flag is set, multiple tables are present in the configuration, and each MMU is dedicated to one of the tables. In this case the MMU translates addresses only when the N/S signal matches the NMS flag. Thus, if there are two tables in the memory management system (one for the system and one for users), the NMS flag is set in those MMUs containing the users' segment descriptors, and is not set in the remaining MMUs. All MMUs in the system have the MST flag set to indicate more than one table in the system.

The final piece of control information in the Mode Register is a 3-bit Identification Field (ID) that indicates a logical name for the MMU. When a segment trap is acknowledged by the CPU, the MMU uses this field to select one of the A/D lines; each enabled MMU should select a different line. If an MMU requested a segment trap, it outputs a 1 on its assigned A/D line; otherwise it outputs a 0. Since the ID field is three bits, up to eight MMUs can be uniquely identified. One instruction might result in multiple violations in different MMUs, so that the segment trap software might have to deal with several MMUs to process the trap.

The other two control registers in the MMU are the Segment Address Register (SAR), which points to one of the 64 segment descriptors, and the Descriptor Selection Counter Register. Commands to read or write a segment descriptor use the SAR pointer to select which descriptor is to be accessed. This register has an auto-incrementing capability for accessing consecutive descriptors in succession without having to reload the SAR. Thus if descriptors 0 through 4 are to be modified, the SAR is initialized to 0 and then auto-incremented to point to descriptors, 1, 2, 3 and 4 in succession.

The Segment Descriptor Number is a 6-bit field that contains the address of the descriptor within the MMU. If the MMU holds segments 64 through 127 (that is, if the URS flag is set), the segment named 64 is accessed when the SAR number field is 0. This is a result of the 6-bit limit of the descriptor number field. The field indicates the 6 least-significant bits of the logical segment descriptor number.

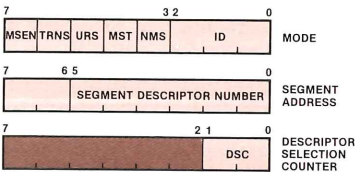


Figure 8. MMU Control Registers

**Control
Registers**
(Continued)

Segment Descriptors consist of four bytes; the Descriptor Selection Counter indicates which byte is being accessed during a command (commands to the MMU can read or write only one byte at a time). A counter value of 0 indicates the high-order byte of the base address is being accessed, 1 indicates the low-order byte of the base address, 2 indicates the limit field, and 3 indicates the attribute field.

**Status
Registers**

Six 8-bit registers contain information useful in recovering from memory trap conditions (Figure 9). The Violation Type Register describes the conditions that generated the segment trap. The Violation Segment Number and Offset Registers contain the segment number and upper byte of the segment address offset for the logical address that caused the segment trap. The Instruction Segment Number and Offset Registers contain the segment number and upper byte of the segment address offset for the last instruction before the segment trap was issued. The Bus Cycle Status Register records the status of the bus at the time the trap condition was detected.

Only violations caused by CPU access have trap information stored in the status registers; DMA violations cause Suppress to be asserted, but the Status Registers are not altered. Thus if a DMA violation occurs between a CPU violation and entry to the trap service routine, the service routine still has the CPU trap information available to process the trap. It is the responsibility of the DMA device to save enough information in the event of a violation so that a software DMA violation service routine can process the violation correctly.

Eight flags in the Violation Type Register describe the cause of the segment trap. Four flags correspond to access protection modes in the segment descriptor attribute mode. A read-only violation sets the RDV flag, a system-only violation sets the SYSV flag, a CPU access to a CPU-Inhibit segment sets the CPUIV flag, an execute-only violation sets the EXCV flag.

Three flags correspond to addressing violation or warnings. The Segment Length Violation Flag (SLV) is set whenever the offset of the logical address falls outside the memory space allocated to the segment. The Primary Write Warning Flag (PWW) is set whenever a write occurs in the last 256 bytes of a segment whose Direction And Warning Flag is set (that is, for segments being used as stacks where the top of the stack is within 256 bytes of the allocated memory space of the segment). The Secondary Write Warning Flag (SWW) is similar to the PWW flag, only it is set when the CPU is in system mode, a stack push is being performed to a segment with a Direction And Warning Flag set, and some other addressing violation or warning has occurred (the EXCV, CPUIV, SLV, SYSV, RDV or PWW flags have been set). When the SWW flag is set it indicates

This counter is used by MMU commands that access multiple bytes within a descriptor. In general, the counter is handled automatically by the MMU commands. Only when a command could be interrupted—and intervening MMU commands issued—should this register be saved and later restored by the interrupting program.

that the system stack is in danger of overflowing its allotted memory. Once the SWW flag is set, further write warnings are suppressed. This prevents the system from repeatedly being interrupted for the same warning while it is in the process of eliminating the cause of the warning.

The final violation-type register flag to be discussed is the Fatal Condition Flag (FATL). This flag is set when any other flag in the violation type register is set and either a violation is detected or a write-warning condition occurs in normal mode. This flag is not set during a stack push in system mode that results in a warning condition. This flag indicates that a memory access error has occurred in the trap processing routine. Once this flag has been set, no Trap Request signals are generated on subsequent violations. However, Suppress signals are generated on this and subsequent CPU violations until the FATL flag has been reset.

The Bus Cycle Status Register contains information pertaining to the status of the bus when a trap condition is detected. This includes CPU Status (ST₀–ST₃), plus flags indicating whether a read or a write was being performed and whether or not the N/S line was asserted.

The Violation Segment Number and Offset Registers record the first logical address to cause a trap. Only the high-order byte of the offset is saved, however, so that external support circuitry is needed to save the low-order eight bits of the logical address offset. If the trap occurred during the instruction fetch cycle, this information is the logical address of the instruction; otherwise it indicates the

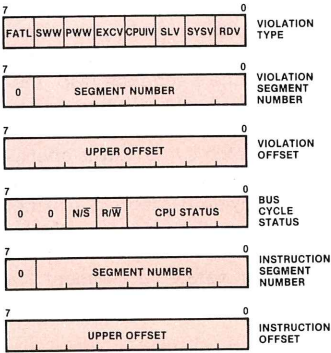


Figure 9. MMU Violation Information Registers

**Status
Registers**
(Continued)

logical address of a data item which was to be accessed.

The Instruction Segment Number and Offset Registers record the logical address of the last instruction fetch that occurred before the trap. Only the high-order byte of the offset is saved, however, so external support circuitry is needed to save the low-order eight bits of the offset.

If an instruction fetch caused the trap, these

registers indicate the logical address of the previous instruction. Such information is useful if the preceding instruction was a branch instruction to an invalid address since—in this case—these registers indicate which branch instruction led to the erroneous situation. If a data reference caused the segment trap, then these registers indicate the logical address of the instruction that specified the illegal access.

**Stack
Segments**

Segments are specified by a base address and a range of legal offsets to this base address. On each access to a segment, the offset is checked against this range to insure that the access falls within the allowed range. If an access outside the segment is attempted, a Trap Request and a Suppress signal are generated.

Normally the legal range of offsets within a segment is from 0 to $256N + 255$ bytes, where $0 \leq N \leq 255$. (N is the value in the limit field of the segment descriptor.) However, a segment may be specified so that legal offsets range from $256N$ to $65,535$ bytes, where $0 \leq N \leq 255$. The latter type of segment is useful for stacks because the Z8001 stack-manipulation instructions cause stacks to grow toward lower memory locations. Thus, when a stack grows to

the limit of its allocated segment, additional memory can be allocated on the correct end of the segment. As an aid in maintaining stacks, the MMU detects when a write is performed to the lowest allocated 256 bytes of these segments and generates a Trap Request. No Suppress signal is generated so the write is allowed to proceed. This write warning can then be used to indicate that more memory should be allocated to the segment.

The DIRW flag indicates that a segment is to be treated in this special way by the MMU. When the DIRW flag is set, the range of allowed offsets is from $256N$ to $65,535$ bytes and writes into the range $256N$ to $256N + 255$ generate Segment Trap but not Suppress, indicating a write warning.

**Segment
Trap and
Acknowledge**

The Z8010 MMU generates a Segment Trap whenever it detects an access violation or a write warning condition. In the case of an access violation, the MMU also activates Suppress. Suppress can be used to inhibit memory writes and to request that special data be returned on a read access. Segment Trap remains Low until a Trap Acknowledge signal is received. If a violation occurs, Suppress is asserted for that cycle and all subsequent CPU memory references until the end of the instruction. Intervening DMA cycles are not suppressed, however, unless they generate a violation. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only; no segment trap requests are ever generated during DMA cycles. This is because the CPU would not be able to respond to these traps until the conclusion of the DMA cycle.

Segment traps to the Z8001 CPU are handled similarly to other types of interrupts. To service a segment trap, the CPU enters a segment trap acknowledge cycle. The acknowledge cycle is always preceded by an instruction fetch cycle that is aborted. The MMU has been designed so that this dummy instruction fetch cycle is ignored. During the acknowledge cycle, all enabled MMUs use the Address/Data lines to indicate their status. An MMU that has generated a Segment Trap request outputs a 1

on the A/D line associated with the number in its ID field. An MMU that has not generated a segment trap request outputs a 0 on its associated A/D line. A/D lines for which no MMU is associated remain 3-stated. During a segment trap acknowledge cycle, an MMU uses A/D line $8 + i$ if the content of its ID field is i .

Following the acknowledge cycle, the CPU automatically pushes the program status words and program counter onto the system stack, and loads a new program status word and program counter from the program status area. The Segment Trap line is reset during the segment trap acknowledge cycle, and no Suppress signal is generated during the stack push. If the store creates a write warning condition, a segment trap request is generated and is serviced at the end of the context swap; the SWW flag is also set. Servicing this second Segment Trap request also creates a write warning condition, but—because the SWW flag is set—no Segment Trap request is generated. If a violation rather than a write warning condition occurs during the context swap, the FATL flag is set rather than the SWW flag. In this case, subsequent violations cause the Suppress to be asserted but not Trap Request. Without the SWW and FATL flags, trap processing routines that generate memory violations would repeatedly be interrupted and called to pro-

**Segment
Trap and
Acknowledge**
(Continued)

cess the violations they create.

The CPU routine to process a trap request should first check the FATL flag to determine if a fatal system error has occurred. If not, the

**Commands
to the MMU**

When a memory management system must read or change information in the MMU to respond to a segment trap or to re-organize the physical memory, it can issue control commands to the MMU. These commands fall into two generic categories: reset commands and read/write commands. Reset commands are simply orders to the MMU to set or clear specified fields. For these commands, the Z8001 Special I/O output command can be used with the destination field set to be the MMU command code corresponding to the desired action.

Read and write commands are slightly more complicated because they consist of both commands and data. Such commands to the MMU are issued using the Z8001 Special I/O instructions. These instructions have a source and a destination field. For an input instruction, the source field contains an MMU command code and the destination field indicates where in primary memory the data is placed. For an output instruction, the destination field contains an MMU command and the source field indicates where the data to be written into the MMU resides in memory.

The high-order byte of the command contains the opcode for that command; the low-order byte of the command can be used to specify the particular MMU to be accessed. The MMU does not receive information on AD₀-AD₇, so external circuitry must decode information on these lines during the Special I/O commands and then select a particular MMU. The encoding of the low-order byte is dependent upon the system implementation. This paper always uses the convention that bit *i* specifies MMU number *i*.

The reset commands to the MMU are: Reset Violation Type Register, Reset SSW Flag In Violation Type Register, and Reset Fatal Flag In Violation Type Register. Resetting the Violation Type Register is similar to a hardware reset in that it clears this register and returns the internal control of the MMU to an initial state (as if no violation had occurred since system initialization). Resetting the SSW flag or the FATL flag in the Violation Type Register clears these flags.

Two other commands are similar to reset commands in that they have no data associated with them. These are Set All CPU-Inhibit Flags in the segment attribute fields and Set All DMA-Inhibit Flags in the segment attribute fields, both of which cause all segment

SSW flag should be checked to determine if more memory is required for the system stack. Finally, the trap itself should be processed and the violation type register reset.

descriptors in the MMU to have the CPUi or DMAi flags set, respectively. These two set commands can be useful in initializing address translation tables or when swapping between tasks. For example, when swapping between tasks the Set All CPUi Flags command automatically makes the previous task's segments inaccessible to the next task, unless the system explicitly initializes the segment attribute field in these segments.

As an example of using the Special Output instruction SOUT to control an MMU, consider resetting the fatal flag of MMU #1. The MMU command opcode for this is "%14" (% denotes hexadecimal). The assembler syntax for the SOUT instruction is "SOUT destination field, source field" so that the instruction to reset the fatal flag of MMU #1 is "SOUT %1402, R0." Specifying register 0 in this instruction is an arbitrary choice—the content of this register is placed on the A/D lines during the data phase of the SOUT instruction, but it is ignored by the MMU. The low-order byte of the command (the destination field of the instruction) encodes which MMU is to reset its fatal flag. The convention followed in this paper is that MMU *i* is specified by setting bit *i* in the low order byte of the command. (Bit 1 set is hex "%02.")

The rest of the MMU commands consist of both operation and data. The following internal registers can be read or written: the Mode Register, the Segment Address Register, the Descriptor Registers and the Descriptor Selection Counter Register. A Descriptor Register can be read or written as a whole, or selected subfields can be accessed. In addition, by using the auto-increment feature of the Segment Address Register, successive Descriptor Registers can be accessed, or a selected field within successive Descriptor Registers can be accessed. For example, one Special I/O command in block mode could read a number of segment attribute fields. This is useful in determining which segments have been modified.

As an example of using the Special Output instruction SOUT to write data into an MMU, consider writing the contents of Register 6 into the Mode Register of MMU #2. The opcode for this command is "%00" and so the command is "SOUT %0004, R6." Here the high-order byte of the destination field contains the opcode and the low-order byte has bit 2 set (hexadecimal 4 if 0100 in binary) indicating MMU #2.

2

Commands to the MMU
(Continued)

Certain MMU internal registers can only be read—there is no corresponding write instruction. This is because these registers contain information relating to a detected violation and thus it is not necessary to be able to write into these registers. These registers are the Violation Type Register, the Violation Segment Number Register, the Violation Offset Register,

the Instruction Segment Number Register, the Instruction Offset Register and the Violation Bus Status Register. Although the Violation Type Register cannot be written, it should be noted that it can be cleared and that two of its flags can be individually cleared: the SWW flag and the FATL flag.

Direct Memory Access

DMA operations may occur between Z8001 machine cycles and can be handled through the MMU. The MMU permits DMA in either the System or Normal Mode of operation. For each memory access, segment attributes are checked and—if a violation is detected—a Suppress signal is generated. Unlike a CPU violation, which automatically causes Suppress signals to be generated on subsequent memory accesses until the next instruction, DMA violations generate a Suppress only on a per-memory-access basis. The DMA device should note the Suppress signal and record sufficient information to enable the system to recover from the access violation. No Segment Trap Request is ever generated during DMA (hence warning conditions are not signaled). There are no trap requests because the CPU would not acknowledge the request until the end of the DMA cycle.

At the start of a DMA cycle, the DMASync line must go Low, indicating to the MMU the beginning of a DMA cycle. A Low DMASync inhibits the MMU from using an indeterminate segment number on lines SN₀–SN₆. When the DMA logical memory address is valid, DMASync must be High on one rising edge of Clock and the MMU then performs its address-translation and access-protection functions. Upon the release of the bus at the termination of the DMA cycle, DMASync must again be High. After two clock cycles of DMASync High, the MMU assumes that the CPU has control of the bus and that subsequent memory references are CPU accesses. The first instruction fetch occurs at least two clock cycles after the CPU regains bus control. During CPU cycles, DMASync should always be High.

Hardware and Software Reset

The MMU can be reset by either hardware or software mechanisms but note that they have different effects. A hardware reset occurs on the falling edge of the Reset input; a software reset is performed by an MMU command. A hardware reset clears the Mode Register, Violation Type Register and Descriptor Selection Counter. If the Chip Select line is Low while Reset is Low the Master Enable Flag in the Mode Register is set to 1. All other registers are undefined. After reset, the A/D and A lines are 3-stated. The SUP and SEGT

open-drain outputs are not driven. If the Master Enable Flag is not set during reset, the MMU does not respond to subsequent addresses on its A/D lines. To enable an MMU after a hardware reset, an MMU command must be used in conjunction with Chip Select.

A software reset occurs when the Reset Violation Type Register command is issued. This command clears the Violation Type Register and returns the MMU to its initial state as if no violations or warnings had occurred.

Multiple-MMU Configurations

Z8010 MMU architecture supports system configurations that use more than one MMU. Multiple MMU devices can be used either to manage 128 CPU segments rather than the 64 supported by one MMU, or to manage multiple translation tables.

The Z8001 CPU generates logical addresses that can specify up to 128 different segment names. Because the MMU contains only 64 Segment Descriptor Registers, two MMUs are needed to perform address translation for 128 logical segments. Systems designed with only one MMU device still have the power and flexibility offered by memory management, although tasks in such a system are restricted to manipu-

lating only 64 logical segment names. These names must either be 0 through 63 or 64 through 127. If the MMU in a single-MMU configuration is set to translate segment names in one range and the CPU generates a logical segment name in the other range, the MMU does not perform address translation and no physical memory location is output. In this case, no request is made to memory. Therefore, a single-MMU configuration should have additional external logic to detect erroneous segment names and generate a Segment Trap and Suppress signal.

The Upper Range Select flag (URS) is used in multiple MMU configurations to indicate which group of logical segment names

Multiple-MMU Configurations

(Continued)

are to be translated by an MMU. When this flag is set, the Segment Descriptor Registers in the MMU are used in translating logical addresses in the range 64 through 127. When the flag is clear, the range is 0 through 63. Thus the URS flag corresponds to the most significant bit (bit 6) in the logical segment names that the MMU translates. Because this flag is under program control, the range of logical segment names can be changed during execution in System Mode.

MMU architecture also supports multiple segment translation tables. This feature is useful when separate tables are maintained for different tasks. Each task has its own table and switching between tasks requires enabling the appropriate MMU devices. In contrast, systems with only one translation table must either restrict the logical segment names that an individual task can use, or change the Descriptor Register entries whenever tasks are swapped. Two flags in the Mode Register, together with the N/S signal, are used in multiple table configurations.

The Multiple Segment Table (MST) flag indicates whether the configuration is being used to support multiple tables. When this flag is set, the MMU will compare the N/S line against the Normal Mode Select Flag (NMS) before generating a physical memory location on its Address lines. When the line and the flag match (both asserted or both de-asserted), the MMU is enabled and an address translation is performed (assuming the URS flag matches the most significant bit in the logical segment

name). If the N/S line fails to match the state of the NMS flag, no translated address is generated by the MMU. The MST flag and the NMS flag are under program control and can be changed in System Mode.

The simplest multiple translation table configuration has one table for Normal Mode access and one for System Mode access. In such a configuration, the Multiple Table Flag is set in all MMUs and the N/S line of each MMU receives its input from the N/S output of the Z8001 CPU. MMUs containing descriptors of system segments have the NMS flag clear, and those containing descriptors to be used in Normal Mode have the flag set. When the Z8001 is in System Mode, the N/S line is Low and it matches the NMS flag in those MMUs whose Descriptor Registers contain system segment information. Therefore, these MMUs are used in address translation for system references.

When the Z8001 is in Normal Mode, the N/S line is High and it matches the NMS flag in those MMUs whose Descriptor Registers contain user segment information. Consequently, these MMUs are used in address translation for user segments. In this configuration, system segments are separated from user segments. When the Z8001 changes from Normal to System Mode of operation, the appropriate translation table is automatically selected. A more elaborate example of a configuration with multiple translation tables is given in the next section.

2

Examples

This section describes two Z8001-Z8010 configurations: one contains two MMUs and one address translation table; the other contains seven MMUs and four address translation tables. These examples are given in sufficient detail to illustrate some of the major ideas in constructing memory-management systems around the Z8010 MMU. High-level block diagrams illustrate some of the major features of typical hardware configurations and short programs illustrate software techniques for using the MMU.

The first example system is the two-MMU configuration illustrated in Figure 10. The two MMUs are called MMU #1 and #2, and they are selected during a command cycle by AD₁ and AD₂ being Low, respectively. Since a Special I/O instruction is being used bit 0 must always be zero. Thus, when a low-order byte of a command is "%02," MMU #1 responds; when it is "%04," MMU #2 responds; and when it is "%06," both MMUs respond. (Note that AD₁ is inverted before attachment to the CS pin.)

The A/D₁ line, which controls MMU #1 through the Chip Select input, is first com-

bined with the Reset line. This allows the Master Enable Flag to be set upon system initialization, so the logical addresses generated by the CPU are passed to the physical memory. This is done because—upon reset—the mode register is otherwise cleared, the Translate Flag is clear and addresses pass through the MMUs untranslated. The bootstrap program can therefore reside in absolute memory locations in the physical memory. If the Reset line is not an input to the Chip Select line, the Master Enable Flag would not be set during system initialization and the CPU would not be able to address memory through the MMUs.

Note that there is a direct path from the CPU and DMA to the system bus. This path is used during I/O and memory refresh because the MMUs are quiescent during these cycles. It is also used for data on memory reads and writes. Also, note that the Suppress line goes both to the memory, where it can be used to protect the memory from erroneous

Examples (Continued)

writes, and back to the DMA device to save information upon the event of a DMA access error.

Of further interest in the example, address latches are used to buffer addresses between the Z8001 and a demultiplexed bus. This is required to demultiplex the address and data onto the bus. The address latch for AD_8-AD_{15} may not be needed if the I/O device does not use separate address and data lines.

A detailed example indicates how such a system could be used. First, consider setting Segment Descriptor Register 65 to point to a read-only segment of 768 bytes starting at memory location %115200. The segment is to be accessed in Normal Mode. The Descriptor Register should be %115202 01. The first two bytes, %1152, indicate the starting location of the segment (note that the low-order byte of the memory address is all zeros and is not stored in the Descriptor Register). The third byte, %02, indicates that three blocks of 256 bytes have been allocated to this segment. The fourth byte, %01, indicates that only the read-only segment flag has been set.

To write this descriptor into the MMU, a copy of the descriptor should be created in primary memory and a Special I/O block transfer instruction used. The SOTIRB instruction can be used for this.

This instruction has the assembler syntax "SOTIRB destination, source, count register" where both the destination and source are registers. The destination register contains the command to the MMU, the memory location pointed to by the source register contains the first byte of the data to be transferred, and the Count Register contains the number of bytes to be transferred.

The opcode to load the Descriptor Register is "%0B". Segment Descriptor Register 65 is Segment Descriptor Register 1 of MMU #2, so the MMU command is "%0B04".

To specify which Segment Descriptor Register to write, it is necessary to load the Segment Address Register of MMU #2 with 1. The MMU opcode to do this is "%01" and so the command is "%0104." The segment number (in this case 65) is a parameter to the example routine, passed in register 0. The

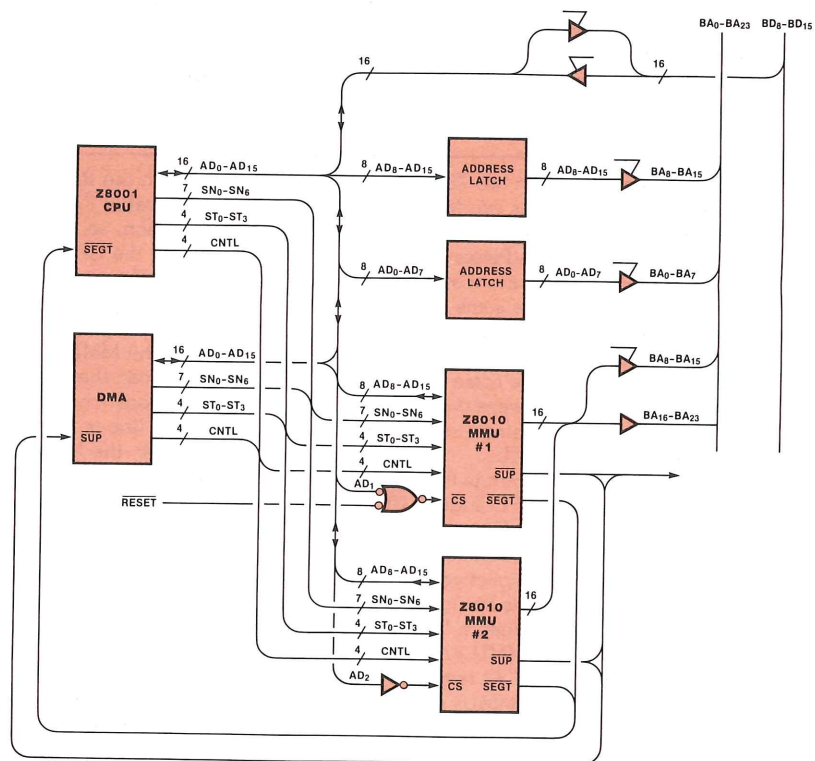


Figure 10. A Dual-MMU Configuration

Examples (Continued)

	BIT	R0, #6	!Test to see if Descriptor Register is in MMU #1!
	JR	Z, OVER	!or MMU #2!
	SOUTB	%0104, R0	!Set SAR in MMU #2!
	LD	R1, #%0B04	!Prepare to write descriptor!
	JR	NEXT	
OVER:	SOUTB	%0102, R0	!Set SAR in MMU #1!
	LD	R1, #%0B02	!Prepare to write descriptor!
NEXT:	LD	R0, #4	!Load count field—4 bytes!
	SOTIRB	@R1, @RR2, R0	!Write descriptor!

descriptor to be written is another parameter to this routine: RR2 contains the address in memory where this information resides. The SOUTB instruction has a similar syntax to the SOTIRB instruction explained previously except that it writes one byte instead of a series of bytes, and the destination I/O address is in the instruction itself instead of in a register specified by the instruction.

The routine on this page initializes the Segment Descriptor. Its parameters are found in Register R0, which contains the segment number to be written, and in Register RR2, which points to the descriptor information in primary memory. Registers R0 through R3 are used by this routine.

Now suppose that the user tries to write into location `< 65 > %9328`. This causes a segment trap both because of the write to a read-only segment and because the access exceeds the segment limit. At the end of the instruction that has the illegal memory access, the CPU acknowledges the trap. During the trap acknowledge cycle, MMU #2 asserts `AD10` (assuming its ID field is "010") and this information is placed on the system stack for the

trap-handling routine.

The trap-handling routine reads the violation information registers from the MMU. The violation type register contains "%05" indicating both a length violation and a read-only violation. The Violation Bus Status Normal Register contains "%28". The first nibble indicates a write in Normal Mode was in progress and the second nibble indicates a memory data access cycle was in progress. The violation segment register contains "%41" indicating segment 1 of MMU #2 caused the violation (which is segment number 65), and the violation offset register contains "%93" indicating the high-order byte of the logical address offset. The operating system can then issue an error message to the user indicating a read-only violation to segment 65. Using the program counter that was stacked when the segment trap was acknowledged, the system can also indicate the next instruction that was to be executed. Note that in this system the low-order byte of the violation offset is lost. This condition is corrected in the next example system.

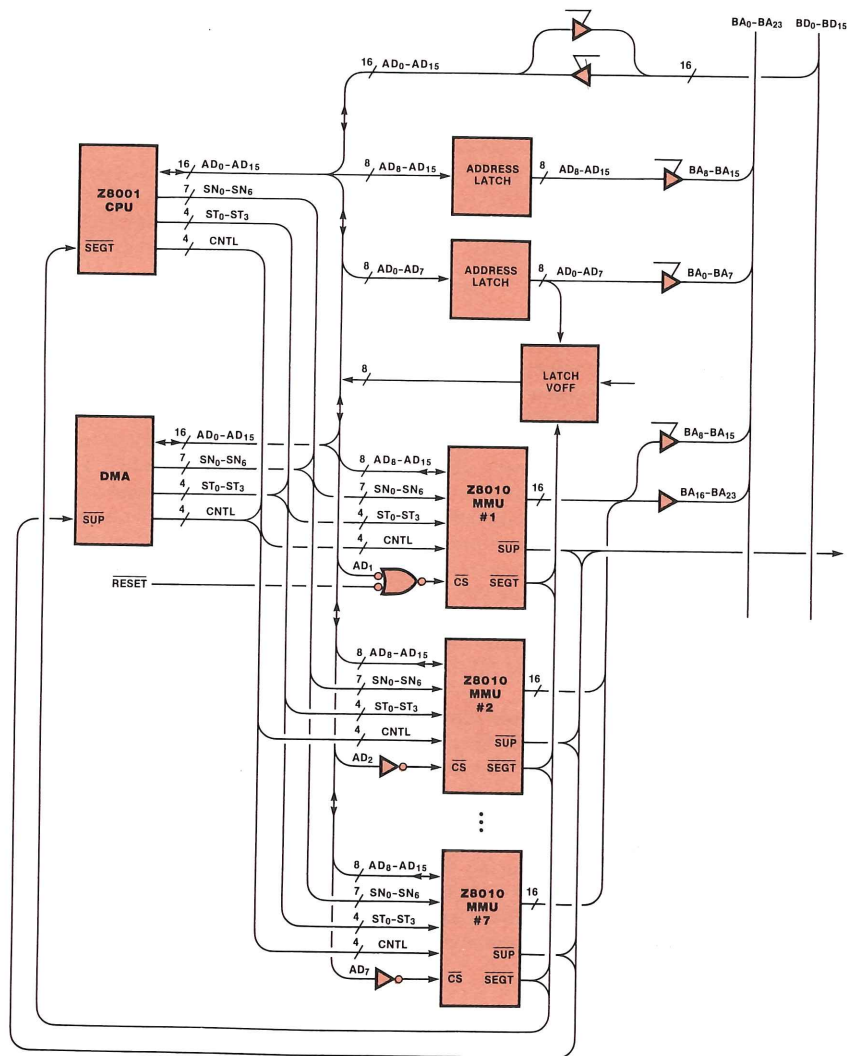


Figure 11. A Seven-MMU Configuration

Figure 11 gives high-level diagram of the second system to be discussed. This configuration contains seven MMUs, and the A/D lines are used directly to select the appropriate MMU when in Command Mode. The major innovation in this example, aside from more MMUs, is the latch to retain the least significant byte of an address offset when a violation is detected. This latch is enabled whenever a segment trap is generated by an MMU and holds the low-order byte of the address that generates an access violation.

Seven MMUs is the limit in one configuration without additional decoding logic for selecting one MMU Chip Select line. The reason why AD₀ cannot be used to control an eighth MMU is due to the read convention of the CPU. When the CPU reads a byte of information and AD₀ is asserted during the address cycle, the data on AD₀-AD₇ is read. However, MMUs put data on AD₈-AD₁₅ during a read command. Thus an MMU that is controlled with AD₀ asserted can never have one of its registers read by the CPU.

Examples (Continued)

The second hardware configuration can support a memory management system designed with one MMU permanently allotted to the operating system (say MMU #1) and the others allotted in pairs to different user tasks. Thus three user tasks can have translation tables resident in the MMUs and switching between active tasks requires that the appropriate MMUs be enabled and disabled. This selection process can be effected by manipulating the Master Enable Flag in the Mode Register of the appropriate MMUs. Note that the system MMU #1 always is enabled.

```

LD      R0,#0
SOUT    %00FC,R0      !Disable all user MMUs by resetting their mode registers!
BIT      R1,#1         !Branch to appropriate enabling code!
JR      NZ,USR2
BIT      R1,#2
JR      NZ,USR3
USR1:   LDB      RL1,#%DA      !Set up bit pattern for mode register; ID=010!
        SOUTB    %0004,RL1     !Store into MMU #2!
        LDB      RL1,#%FB      !ID=011, URS Flag=1!
        SOUTB    %0008,RL1     !Store into MMU#3!
        JR      END
USR2:   LDB      RL1,#%DC
        SOUTB    %0010,RL1
        LDB      RL1,#%FD
        SOUTB    %0020,RL1
        JR      END
USR3:   LDB      RL1,#%DE
        SOUTB    %0040,RL1
        LDB      RL1,#%FF
        SOUTB    %0080,RL1
END:

```

The software routine on this page performs the selective enabling of MMUs required by a task swap. This routine disables all user MMUs (which thereby also disables the currently enabled user MMUs) and then enables the appropriate pair. The input to this routine is a code for the active task that is to be dispatched next. The code is in register R1; bit #0 is set if the first task is to be dispatched, bit #1 is set if the second task is to be dispatched and bit #2 is set if the third task is to be dispatched.

2

MMU Command Summary

Opcode	Operation	Opcode	Operation
00	Read/Write Mode Register	0C	Read/Write Base Field And Increment SAR
01	Read/Write Segment Address Register	0D	Read/Write Limit Field And Increment SAR
02	Read Violation Type Register	0E	Read/Write Attribute Field And Increment SAR
03	Read Violation Segment Number	0F	Read/Write Descriptor And Increment SAR
04	Read Violation Offset (high byte)	10	Reserved
05	Read Bus Cycle Status Register	11	Reset Violation Type Register
06	Read Instruction Segment Number	12	Reserved
07	Read Instruction Offset (high byte)	13	Reset SWW Flag In VTR
08	Read/Write Base Field In Descriptor	14	Reset FATL Flag In VTR
09	Read/Write Limit Field In Descriptor	15	Set All CPU-Inhibit Flags
0A	Read/Write Attribute Field In Descriptor	16	Set All DMA-Inhibit Flags
0B	Read/Write Descriptor (all fields)	17-1F	Reserved
		20	Read/Write Descriptor Selector Counter Register
		21-3F	Reserved

Description (Continued)

Z8 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a

microprocessor that can address 124K of external memory.

The Z8 offers three basic address spaces to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, 4 I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, the Z8 offers an on-chip asynchronous receiver/transmitter (UART), and two counter/timers with a large number of user-selectable modes. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

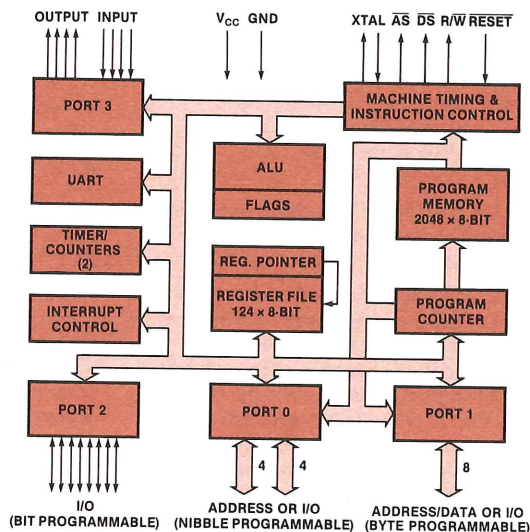


Figure 3. Functional Block Diagram

Pin Description

P0₀-P0₇, P1₀-P1₇, P2₀-P2₇, P3₀-P3₇. I/O Port Lines (Input/Outputs, TTL compatible). These 32 lines are divided into four 8-bit I/O ports that can be configured under program control for I/O or external memory interface.

AS. Address Strobe (output, active Low). Address Strobe is pulsed once for internal and external program fetches and external data memory transfers. The addresses for all external program or data memory transfers are valid at the trailing edge of AS. Under pro-

gram control, AS can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

DS. Data Strobe (output, active Low). Data Strobe is activated once for each external memory transfer. When the Z8 is not configured for external memory transfer, DS acts as an instruction sync signal and is forced Low during the internal clock period preceding the beginning of the opcode fetch.

**Pin
Description**
(Continued)

R/ $\overline{W}$. Read/Write (output). $R/\overline{W}$ is Low when the Z8 is writing to external program or data memory.

XTAL1, XTAL2. Crystal 1, Crystal 2 (time-base input and output). These pins connect a series-resonant crystal (8 MHz maximum), LC network, RC network or an external single-phase

clock (8 MHz maximum) to the on-chip clock oscillator and buffer.

RESET. Reset (input, active Low). $\overline{RESET}$ initializes the Z8. When $\overline{RESET}$ is deactivated, the Z8 begins program execution from internal program location 000CH.

Architecture

Z8 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a

microprocessor that can address 124K of external memory.

The Z8 offers three basic address spaces to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, 4 I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, the Z8 offers an on-chip asynchronous receiver/transmitter (UART), and two counter/timers with a large number of user-selectable modes. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

2

**Address
Spaces**

Program Memory. The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 2048 bytes consist of on-chip mask-programmed ROM. At addresses 2048 and greater, the Z8 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These loca-

tions contain six 16-bit vectors that correspond to the six available interrupts.

Data Memory. The Z8 can address 62K bytes of external data memory beginning at locations 2048 (Figure 5). External data memory may be included with or separated from the external program memory space. DM, an optional I/O function that can be programmed to appear on pin P3₄, is used to distinguish between data and program memory space.

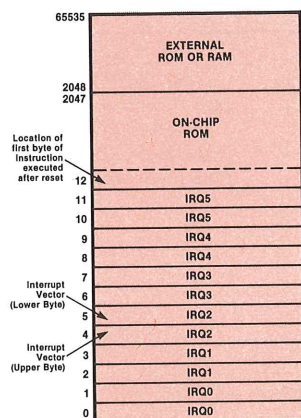


Figure 4. Program Memory Map

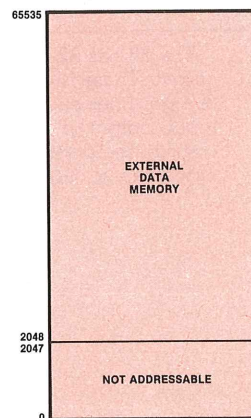


Figure 5. Data Memory Map

Register File. The 144-byte register file includes four I/O port registers (R0–R3), 124 general-purpose registers (R4–R127) and sixteen control and status registers (R240–R255). These registers are assigned the address locations shown in Figure 6.

Z8 instructions can access registers directly or indirectly with an 8-bit address field. The Z8 also allows fast 4-bit register addressing using the register pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying sixteen contiguous locations

(Figure 7). The register pointer addresses the starting location of the active working-register group.

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit stack pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 2048 and 65535. An 8-bit stack pointer (R255) is used for the internal stack which resides within the 124 general-purpose registers (R4–R127).

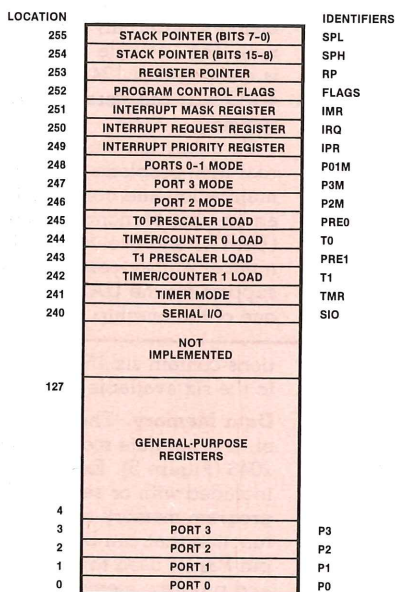


Figure 6. The Register File

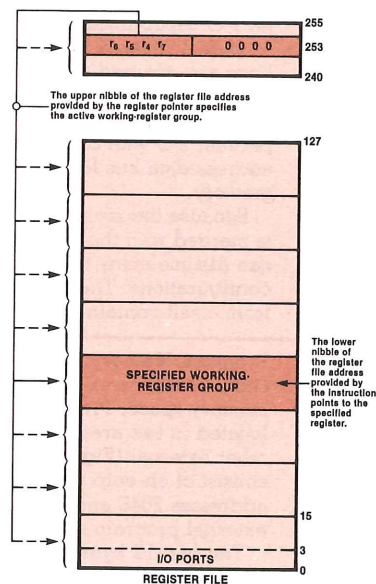


Figure 7. The Register Pointer

I/O Ports

The Z8 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals,

serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

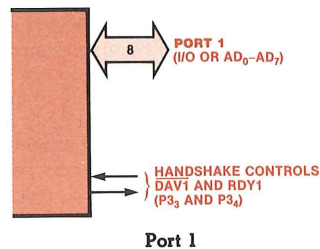
In the following I/O port discussions, Port 1 is discussed before Port 0 because Port 1 outputs the lower eight address bits.

I/O Ports (Continued)

Port 1 can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines P₃ and P₄ are used as the handshake controls RDY1 and DAV1 (Ready and Data Available).

Memory locations greater than 2048 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed address/data mode (AD₀–AD₇). If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, $\overline{AS}$, $\overline{DS}$ and R/W,

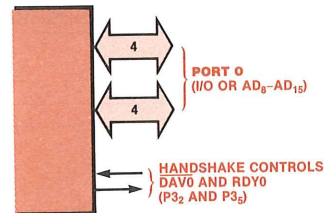


Port 1

allowing the Z8 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning P₃ as a Bus Acknowledge input, and P₄ as a Bus Request output.

Port 0 can be programmed as a nibble I/O port, or as an address/data port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines P₃ and P₅ are used as the handshake controls DAV0 and RDY0.

For external memory references, Port 0 can provide address bits A₈–A₁₁ (lower nibble) or A₈–A₁₅ (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are

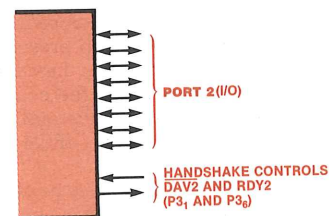


Port 0

defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals $\overline{AS}$, $\overline{DS}$ and R/W.

Port 2 bits can be programmed independently as an input or an output. Because all external memory addresses originate from Ports 0 and 1, Port 2 is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines P₃ and P₆ are used as the handshake controls lines DAV2 and RDY2. The handshake signal assignment for Port 3 lines P₃ and P₆ is dictated by the direction

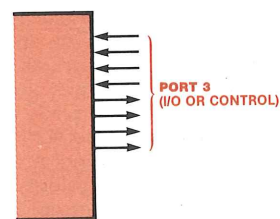


Port 2

(input or output) assigned to bit 7 of Port 2.

Port 3 lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input (P₃–P₆) and four output (P₇–P₁₀). For serial I/O, lines P₃ and P₇ are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 (DAV and RDY); four external interrupt request signals (IRQ0–IRQ3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select (DM).



Port 3

Serial Input/Output

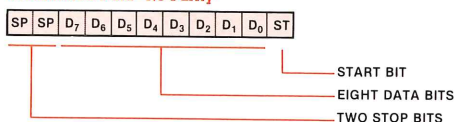
Port 3 lines P3₀ and P3₇ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by counter/timer 0, with a maximum rate of 62.5 kilobits per second.

The Z8 automatically adds a start bit and two stop bits to transmitted data (Figure 8). The Z8 can also provide odd parity. Eight data bits are always transmitted, regardless of parity selec-

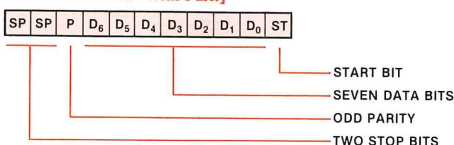
tion. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

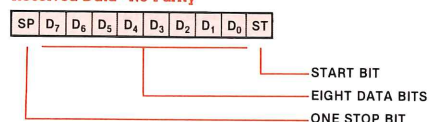
Transmitted Data - No Parity



Transmitted Data - With Parity



Received Data - No Parity



Received Data - With Parity

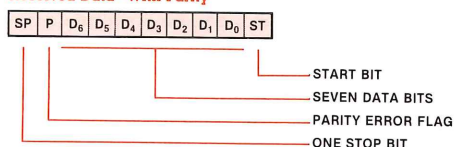


Figure 8. Serial Data Formats

Counter/Timers

The Z8 contains two 8-bit programmable counter/timers (T₀ and T₁), each driven by its own 6-bit programmable prescaler. The T₁ prescaler can be driven by internal or external clock sources; however, the T₀ prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ4 (T₀) or IRQ5 (T₁)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-

pass mode), or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T₁ is user-definable and can be the internal microprocessor clock (4 MHz maximum) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T₀ output to the input of T₁. Port 3 line P3₆ also serves as a timer output (T_{OUT}) through which T₀, T₁ or the internal clock can be output.

Interrupts

The Z8 allows six different interrupts from eight sources: the four Port 3 lines P3₀–P3₃, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask Register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority Register.

All Z8 interrupts are vectored. When an interrupt request is granted, the Z8 enters an interrupt machine cycle that disables all subse-

quent interrupts, saves the program counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

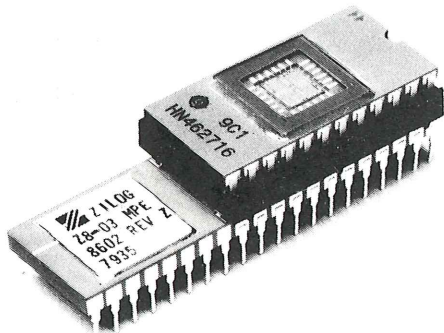
The Z8 also supports polled systems. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request Register polled to determine which of the interrupt requests needs service.

Z8 Protopack Emulator

The Z8-03 MPE (Z8 PROTOPACK) is used for prototype development and preproduction of mask-programmed Z8-based applications. The Z8 Protopack is the ROM-less version of the standard Z8, housed in a pin-compatible 40-pin package.

To provide pin compatibility and interchangeability with the standard mask-programmed Z8, the Z8 Protopack carries (piggybacks) a 24-pin socket for a direct interface to program memory (Figure 1). The 24-pin socket is equipped with 12 ROM Address lines, 8 ROM Data lines and necessary control lines for interface to 2716 EPROM for the first $2K \times 8$ bytes of program memory.

Pin compatibility allows the user to design the pc board for a final 40-pin mask-programmed Z8, and, at the same time, allows



The Z8 Microcomputer Protopack Emulator

the use of a Z8 Protopack to build the prototype and pilot production units. When the final program is established, the user can then switch over to the 40-pin mask programmed Z8 for large volume production. The new Z8 Protopack is also useful in small volume applications where masked ROM setup time, mask charges, etc., are prohibitive and program flexibility is desired.

Compared to the conventional EPROM versions of the single-chip microcomputers, the Z8 Protopack approach offers two main advantages:

- Ease of developing various programs during the prototyping stage: For instance, in applications where the same hardware configuration is used with more than one program, the Z8 Protopack allows economical program storage in separate EPROMs (or PROMs), whereas the use of separate EPROM-based single-chip microcomputers is more costly.
- Elimination of long lead time in procuring EPROM-based microcomputers.

Highly adaptable to user needs, all versions of the Z8 offer high-speed instruction execution, efficient memory use, sophisticated interrupt and bit-manipulation capabilities, and system expandability. The device operates from a single +5 V supply. The standard 40-pin Z8 offers $2K \times 8$ ROM, 144×8 RAM, 32 I/O lines, two counter/timers and one serial I/O port.

Z8/64 Development Device

The 64-pin development version of the 40-pin mask-programmed Z8 allows the user to prototype the system in hardware with an actual Z8 device, and develop the code that is eventually mask-programmed into the on-chip ROM of the Z8/40.

The Z8/64 is identical to the Z8/40 with the following exceptions:

- The internal ROM has been removed.
- The ROM address lines and data lines are buffered and brought out to external pins.
- Control lines for the new memory have been added.

Pin Description

The functions of the Z8/64 I/O lines, $\overline{AS}$, $\overline{DS}$, R/W , XTAL1, XTAL2 and RESET are identical to those of their Z8/40 counterparts. The functions of the remaining 24 pins are as follows:

A₀-A₁₁. Program Memory Address (outputs). A₀-A₁₁ access the first 2K bytes of program memory. A₁₁ is a reserved pin.

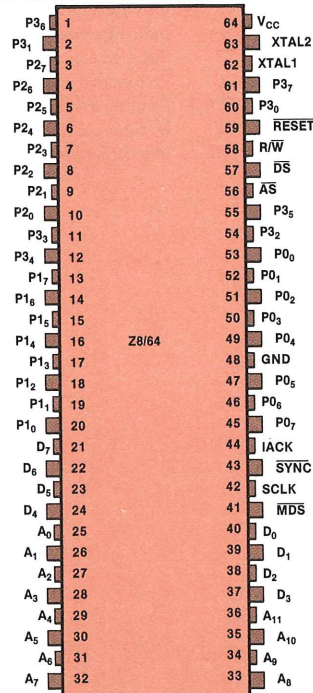


Figure 9. Z8/64 Pin Assignments

D₀-D₇. *Program Data* (inputs). Program data from the first 2K bytes of program memory is input through pins D₀-D₇.

MDS. *Program Memory Data Strobe* (output, active Low). MDS is Low during an instruction fetch cycle when the first 2K bytes of program memory are being accessed.

SYNC. *Instruction Sync* (output, active Low). This strobe output is forced Low during the

internal clock period preceding the beginning of an opcode fetch.

SCLK. *System Clock* (output). SCLK is the internal clock output through a buffer. The clock rate is equal to one-half the crystal frequency.

IACK. *Interrupt Acknowledge* (output, active High). IACK is driven High in response to an interrupt during the interrupt machine cycle.

Z8 Instruction Set Notation

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "**←**". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

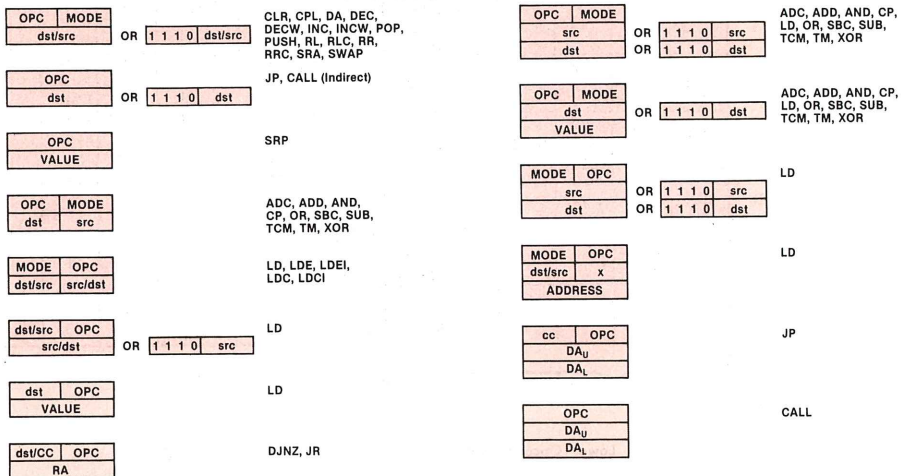
0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
—	Unaffected
X	Undefined

Condition Codes	Value	Mnemonic	Meaning	Flags Set
	1000		Always true	---
	0111	C	Carry	C = 1
	1111	NC	No carry	C = 0
	0110	Z	Zero	Z = 1
	1110	NZ	Not zero	Z = 0
	1101	PL	Plus	S = 0
	0101	MI	Minus	S = 1
	0100	OV	Overflow	V = 1
	1100	NOV	No overflow	V = 0
	0110	EQ	Equal	Z = 1
	1110	NE	Not equal	Z = 0
	1001	GE	Greater than or equal	(S XOR V) = 0
	0001	LT	Less than	(S XOR V) = 1
	1010	GT	Greater than	[Z OR (S XOR V)] = 0
	0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
	1111	UGE	Unsigned greater than or equal	C = 0
	0111	ULT	Unsigned less than	C = 1
	1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
	0011	ULE	Unsigned less than or equal	(C OR Z) = 1
	0000		Never true	---

Instruction Formats



One-Byte Instructions



Two-Byte Instructions

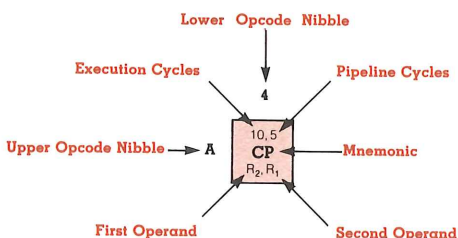
Three-Byte Instructions

Figure 10. Instruction Formats

Z8 Opcode Map

		Lower Nibble (Hex)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Upper Nibble (Hex)	0	6,5 DEC R ₁	6,5 DEC IR ₁	6,5 ADD r ₁ , r ₂	6,5 ADD r ₁ , Ir ₂	10,5 ADD R ₂ , R ₁	10,5 ADD IR ₂ , R ₁	10,5 ADD R ₁ , IM	10,5 ADD IR ₁ , IM	6,5 LD r ₁ , R ₂	6,5 LD r ₂ , R ₁	12/10,3/5 DJNZ r ₁ , RA	12/10,3 JR cc, RA	6,5 LD r ₁ , IM	12/10,0 JP cc, DA	6,5 INC r ₁	
	1	6,5 RLC R ₁	6,5 RLC IR ₁	6,5 ADC r ₁ , r ₂	6,5 ADC r ₁ , Ir ₂	10,5 ADC R ₂ , R ₁	10,5 ADC IR ₂ , R ₁	10,5 ADC R ₁ , IM	10,5 ADC IR ₁ , IM								
	2	6,5 INC R ₁	6,5 INC IR ₁	6,5 SUB r ₁ , r ₂	6,5 SUB r ₁ , Ir ₂	10,5 SUB R ₂ , R ₁	10,5 SUB IR ₂ , R ₁	10,5 SUB R ₁ , IM	10,5 SUB IR ₁ , IM								
	3	8,0 JP IRR ₁	6,1 SRP IM	6,5 SBC r ₁ , r ₂	6,5 SBC r ₁ , Ir ₂	10,5 SBC R ₂ , R ₁	10,5 SBC IR ₂ , R ₁	10,5 SBC R ₁ , IM	10,5 SBC IR ₁ , IM								
	4	8,5 DA R ₁	8,5 DA IR ₁	6,5 OR r ₁ , r ₂	6,5 OR r ₁ , Ir ₂	10,5 OR R ₂ , R ₁	10,5 OR IR ₂ , R ₁	10,5 OR R ₁ , IM	10,5 OR IR ₁ , IM								
	5	10,5 POP R ₁	10,5 POP IR ₁	6,5 AND r ₁ , r ₂	6,5 AND r ₁ , Ir ₂	10,5 AND R ₂ , R ₁	10,5 AND IR ₂ , R ₁	10,5 AND R ₁ , IM	10,5 AND IR ₁ , IM								
	6	6,5 COM R ₁	6,5 COM IR ₁	6,5 TCM r ₁ , r ₂	6,5 TCM r ₁ , Ir ₂	10,5 TCM R ₂ , R ₁	10,5 TCM IR ₂ , R ₁	10,5 TCM R ₁ , IM	10,5 TCM IR ₁ , IM								
	7	10/12,1 PUSH R ₂	12/14,1 PUSH IR ₂	6,5 TM r ₁ , r ₂	6,5 TM r ₁ , Ir ₂	10,5 TM R ₂ , R ₁	10,5 TM IR ₂ , R ₁	10,5 TM R ₁ , IM	10,5 TM IR ₁ , IM								
	8	10,5 DECW RR ₁	10,5 DECW IR ₁	12,0 LDE r ₁ , Ir ₂	18,0 LDEI Ir ₁ , Ir ₂												6,1 DI
	9	6,5 RL R ₁	6,5 RL IR ₁	12,0 LDE r ₂ , Ir ₁	18,0 LDEI Ir ₂ , Ir ₁												6,1 EI
	A	10,5 INCW RR ₁	10,5 INCW IR ₁	6,5 CP r ₁ , r ₂	6,5 CP r ₁ , Ir ₂	10,5 CP R ₂ , R ₁	10,5 CP IR ₂ , R ₁	10,5 CP R ₁ , IM	10,5 CP IR ₁ , IM								14,0 RET
	B	6,5 CLR R ₁	6,5 CLR IR ₁	6,5 XOR r ₁ , r ₂	6,5 XOR r ₁ , Ir ₂	10,5 XOR R ₂ , R ₁	10,5 XOR IR ₂ , R ₁	10,5 XOR R ₁ , IM	10,5 XOR IR ₁ , IM								16,0 IRET
	C	6,5 RRC R ₁	6,5 RRC IR ₁	12,0 LDC r ₁ , Ir ₂	18,0 LDCI Ir ₁ , Ir ₂				10,5 LD r ₁ , x, R ₂								6,5 RCF
	D	6,5 SRA R ₁	6,5 SRA IR ₁	12,0 LDC r ₂ , Ir ₁	18,0 LDCI Ir ₂ , Ir ₁	20,0 CALL* IRR ₁		20,0 CALL DA	10,5 LD r ₂ , x, R ₁								6,5 SCF
	E	6,5 RR R ₁	6,5 RR IR ₁		6,5 LD r ₁ , Ir ₂	10,5 LD R ₂ , R ₁	10,5 LD IR ₂ , R ₁	10,5 LD R ₁ , IM	10,5 LD IR ₁ , IM								6,5 CCF
	F	8,5 SWAP R ₁	8,5 SWAP IR ₁		6,5 LD Ir ₁ , r ₂		10,5 LD R ₂ , R ₁										6,0 NOP

Bytes Per Instruction



Legend:

R = 8-Bit Address
r = 4-Bit Address
R₁ or r₁ = Dst Address
R₂ or r₂ = Src Address

Sequence:

Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction; fetch cycle appears as a 3-byte instruction.

Instruction Summary

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
ADC dst,src dst ← dst + src + C	(Note 1)		1□	*	*	*	*	0	*	
ADD dst,src dst ← dst + src	(Note 1)		0□	*	*	*	*	0	*	
AND dst,src dst ← dst AND src	(Note 1)		5□	-	*	*	0	-	-	
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA	IRR	D6 D4	-	-	-	-	-	-	
CCF C ← NOT C			EF	*	-	-	-	-	-	
CLR dst dst ← 0	R	IR	B0 B1	-	-	-	-	-	-	
COM dst dst ← NOT dst	R	IR	60 61	-	*	*	0	-	-	
CP dst,src dst ← src	(Note 1)		A□	*	*	*	*	-	-	
DA dst dst ← DA dst	R	IR	40 41	*	*	*	X	-	-	
DEC dst dst ← dst - 1	R	IR	00 01	-	*	*	*	-	-	
DECW dst dst ← dst - 1	RR	IR	80 81	-	*	*	*	-	-	
DI IMR (7) ← 0			8F	-	-	-	-	-	-	
DINZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	-	-	-	-	-	-	
EI IMR (7) ← 1			9F	-	-	-	-	-	-	
INC dst dst ← dst + 1	r		rE r=0-F	-	*	*	*	-	-	
	R	IR	20 21							
INCW dst dst ← dst + 1	RR	IR	A0 A1	-	*	*	*	-	-	
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	*	*	*	*	*	*	
JP cc,dst if cc is true PC ← dst	DA	IRR	cD c=0-F	-	-	-	-	-	-	
			30							
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	-	-	-	-	-	-	
LD dst,src dst ← src	r	IM	rC	-	-	-	-	-	-	
	r	R	r8							
	R	r	r9							
			r=0-F							
	r	X	C7							
	X	r	D7							
	r	Ir	E3							
	Ir	r	F3							
	R	R	E4							
	R	IR	E5							
	R	IM	E6							
	IR	IM	E7							
	IR	R	F5							
LDC dst,src dst ← src	r	Irr	C2 D2	-	-	-	-	-	-	
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir	Irr	C3 D3	-	-	-	-	-	-	
LDE dst,src dst ← src	r	Irr	82 92	-	-	-	-	-	-	
	Irr	r								

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir	Irr	83 93	-	-	-	-	-	-	
NOP			FF	-	-	-	-	-	-	
OR dst,src dst ← dst OR src	(Note 1)		4□	-	*	*	0	-	-	
POP dst dst ← @SP SP ← SP + 1	R	IR	50 51	-	-	-	-	-	-	
PUSH src SP ← SP - 1; @SP ← src	R	IR	70 71	-	-	-	-	-	-	
RCF C ← 0			CF	0	-	-	-	-	-	
RET PC ← @SP; SP ← SP + 2			AF	-	-	-	-	-	-	
RL dst		R	IR	90 91	*	*	*	*	-	
RLC dst		R	IR	10 11	*	*	*	*	-	
RR dst		R	IR	E0 E1	*	*	*	*	-	
RRC dst		R	IR	C0 C1	*	*	*	*	-	
SBC dst,src dst ← dst - src - C	(Note 1)		3□	*	*	*	*	1	*	
SCF C ← 1			DF	1	-	-	-	-	-	
SRA dst		R	IR	D0 D1	*	*	*	0	-	
SRP src RP ← src		IM	31	-	-	-	-	-	-	
SUB dst,src dst ← dst - src	(Note 1)		2□	*	*	*	*	1	*	
SWAP dst		R	IR	F0 F1	X	*	*	X	-	
TCM dst,src (NOT dst) AND src	(Note 1)		6□	-	*	*	0	-	-	
TM dst,src	(Note 1)		7□	-	*	*	0	-	-	
XOR dst,src dst ← dst XOR src	(Note 1)		B□	-	*	*	0	-	-	

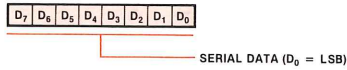
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity in this table. The higher opcode nibble is found in the instruction set table above. The lower nibble is expressed symbolically by a □ in the table above, and its value is found in the following table to the right of the applicable addressing mode pair.

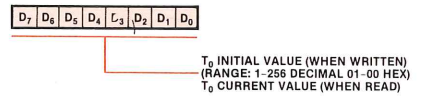
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode		Lower Opcode Nibble
dst	src	
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

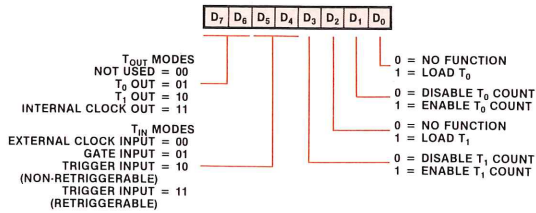
R240 SIO Serial I/O Register (F0_H; Read/Write)



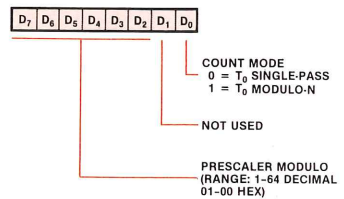
R244 T0 Counter/Timer 0 Register (F4_H; Read/Write)



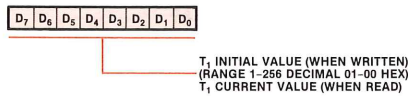
R241 TMR Timer Mode Register (F1_H; Read/Write)



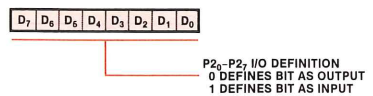
R245 PRE0 Prescaler 0 Register (F5_H; Write Only)



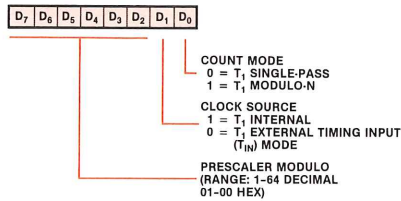
R242 T1 Counter Timer 1 Register (F2_H; Read/Write)



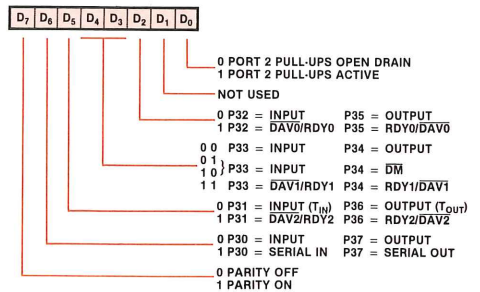
R246 P2M Port 2 Mode Register (F6_H; Write Only)



R243 PRE1 Prescaler 1 Register (F3_H; Write Only)

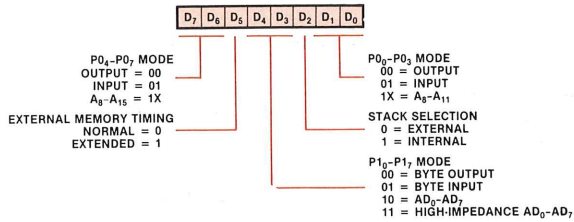


R247 P3M Port 3 Mode Register (F7_H; Write Only)

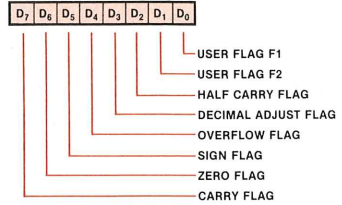


Z8 Control Registers

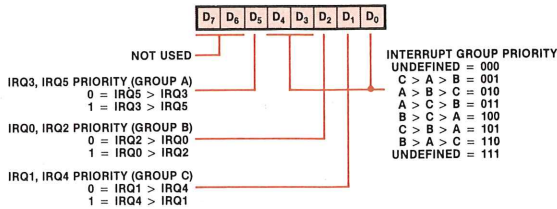
R248 P01M Port 0 and 1 Mode Register (F8_H; Write Only)



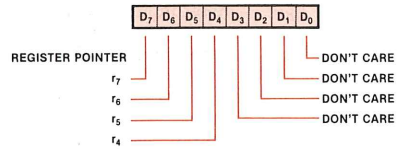
R252 FLAGS Flag Register (FC_H; Read/Write)



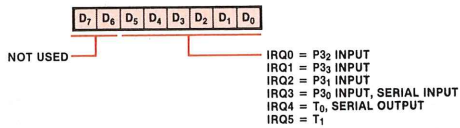
R249 IPR Interrupt Priority Register (F9_H; Write Only)



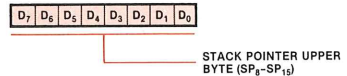
R253 RP Register Pointer (FD_H; Read/Write)



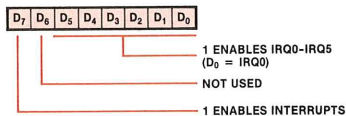
R250 IRQ Interrupt Request Register (FA_H; Read/Write)



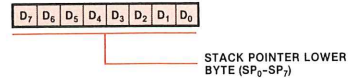
R254 SPH Stack Pointer (FE_H; Read/Write)



R251 IMR Interrupt Mask Register (FB_H; Read/Write)



R255 SPL Stack Pointer (FF_H; Read/Write)



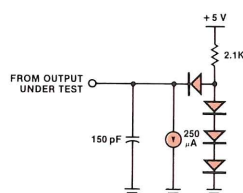
Z8 Control Registers

Absolute Maximum Ratings	Voltages on all inputs and outputs with respect to GND -0.3 V to +7.0 V Operating Ambient Temperature 0°C to +70°C Storage Temperature -65°C to +150°C	Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.
Standard Test Conditions	The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:	$\square +4.75\text{ V} \leq V_{CC} \leq +5.25\text{ V}$ $\square \text{GND} = 0\text{ V}$ $\square 0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$

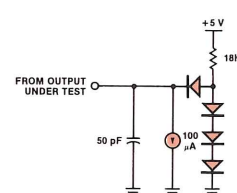
DC Characteristics	Symbol	Parameter	Min	Max	Unit	Condition	Notes
	V_{CH}	Clock Input High Voltage			V	Driven by External Clock Generator	
	V_{CL}	Clock Input Low Voltage			V	Driven by External Clock Generator	
	V_{IH}	Input High Voltage	2.0	V_{CC}	V		
	V_{IL}	Input Low Voltage	-0.3	0.8	V		
	V_{RH}	Reset Input High Voltage			V		
	V_{RL}	Reset Input Low Voltage			V		
	V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\text{ }\mu\text{A}$	1
	V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0\text{ mA}$	1
	I_{IL}	Input Leakage			μA	$0 \leq V_{IN} \leq +5.25\text{ V}$	
	I_{OL}	Output Leakage			μA	$0 \leq V_{IN} \leq +5.25\text{ V}$	
	I_{IR}	Reset Input Current			μA	$V_{CC} = +5.25\text{ V}, V_{RL} = 0\text{ V}$	
	I_{CC}	V_{CC} Supply Current			mA		
	I_{MM}	V_{MM} Supply Current			mA		

1. For A_0 - A_{11} , $\overline{\text{MDS}}$, $\overline{\text{SYNC}}$, SCLK and IACK on the Z8/64 pin version, $I_{OH} = -100\text{ }\mu\text{A}$ and $I_{OL} = 1.0\text{ mA}$.

Test Load Circuits



Test Load 1

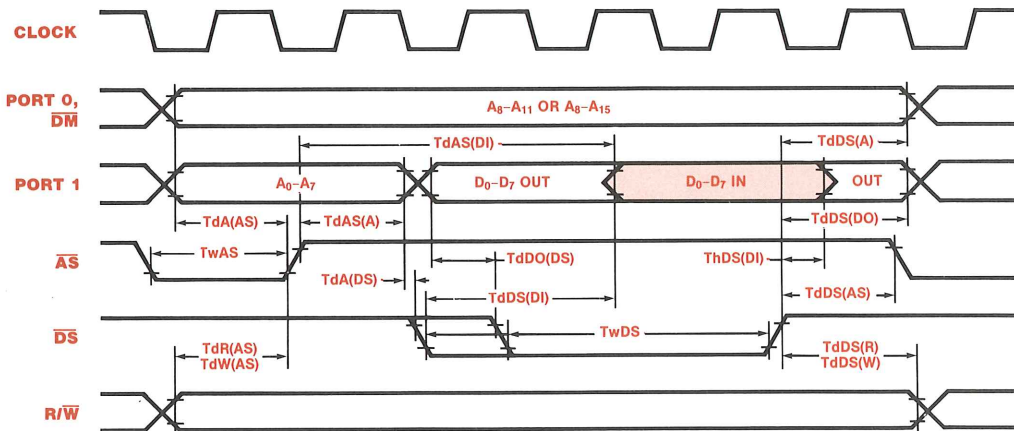


Test Load 2

**External I/O
or Memory
Read and
Write Cycle**

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay	30		ns	Test Load 1	1
TdAS(A)	Address Strobe to Address Float Delay	60		ns	Test Load 1	1
TdAS(DI)	Address Strobe to Data In Valid Delay		280	ns	Test Load 1	3
TwAS	Address Strobe Width	60		ns	Test Load 1	1
TdA(DS)	Address Float to Data Strobe Delay	0		ns	Test Load 1	
TwDS	Data Strobe Width	230		ns	Test Load 1	2
TdDS(DI)	Data Strobe to Data In Valid Delay		160	ns	Test Load 1	3
ThDS(DI)	Data In Hold Time	0		ns		
TdDS(A)	Data Strobe to Address Change Delay	60		ns	Test Load 1	1
TdDS(AS)	Data Strobe to Address Strobe Delay	50		ns	Test Load 1	1
TdR(AS)	Read Valid to Address Strobe Delay	30		ns	Test Load 1	1
TdDS(R)	Data Strobe to Read Change Delay	60		ns	Test Load 1	1
TdDO(DS)	Data Out Valid to Data Strobe Delay	30		ns	Test Load 1	1
TdDS(DO)	Data Strobe to Data Out Change Delay	60		ns	Test Load 1	1
TdW(AS)	Write Valid to Address Strobe Delay	30		ns	Test Load 1	1
TdDS(W)	Data Strobe to Write Change Delay	60		ns	Test Load 1	1

1. Delay times are specified for an input clock frequency of 8 MHz. When operating at a lower frequency, the increase in input clock period must be added to the specified delay time.
2. Data Strobe Width is specified for an input clock frequency of 8 MHz. When operating at a lower frequency, the increase in three input clock periods must be added to the specified width. Data Strobe Width varies according to the instruction being executed.
3. Address Strobe and Data Strobe to Data In Valid delay times represent memory system access times and are given for an 8 MHz crystal input frequency. For lower frequencies; the change in four clock periods must be added to TdAS(DI) and the change in three clock periods added to TdDS(DI).
4. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0."



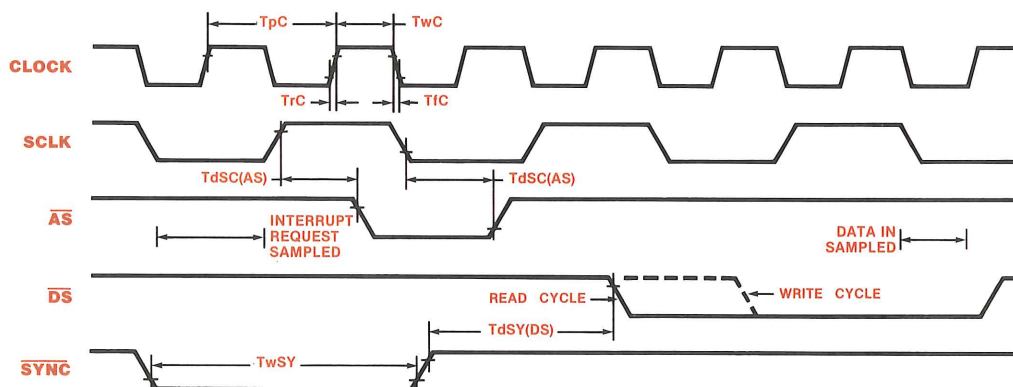
Additional Timing	Symbol	Parameter	Min	Max	Unit	Condition	Notes
	TpC	Input Clock Period	125		ns		
	TrC, TfC	Input Clock Rise and Fall Times			ns	From External Clock Generator	
	TwC	Input Clock Width			ns	From External Clock Generator	
	TdSC(AS)	System Clock Out to Address Strobe Delay Time			ns		1
	TdSY(DS)	Instruction Sync Out to Data Strobe Delay Time			ns		1, 2
	TwSY	Instruction Sync Out Width			ns		1, 2

1. Test Conditions use Test Load 1 for SCLK and SYNC when output through their respective Port 3 pins and Test Load 2 on the SCLK and SYNC direct outputs on the 64-pin version.

2. Times given assume an 8 MHz crystal input frequency. For

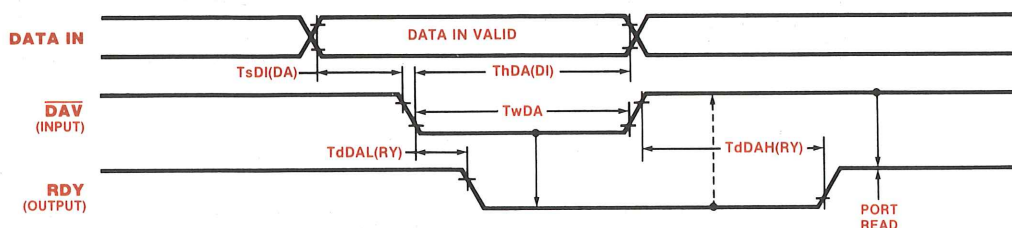
lower frequencies, the change in two clock periods must be added.

3. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0."

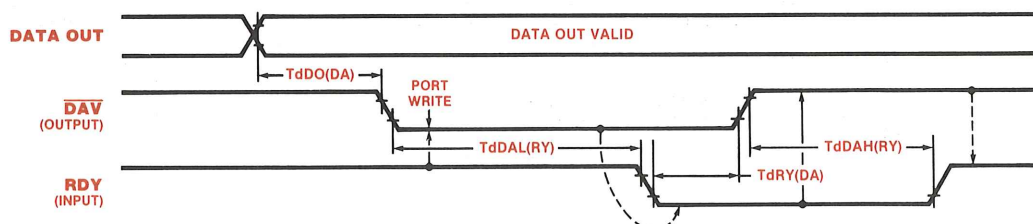


Handshake Timing	Symbol	Parameter	Min	Max	Unit	Condition
	TsDI(DA)	Data In Setup Time	0		ns	
	ThDA(DI)	Data In Hold Time	190		ns	
	TwDA	Data Available Width			ns	Input Handshake Test Load 1
	TdDAL(RY)	Data Available Low to Ready Delay Time	0		ns	Input Handshake Test Load 1
	TdDAH(RY)	Data Available High to Ready Delay Time	0		ns	Output Handshake Test Load 1
	TdDO(DA)	Data Out to Data Available Delay Time			ns	Test Load 1
	TdRY(DA)	Ready to Data Available Delay Time			ns	Test Load 1

2



Input Handshake

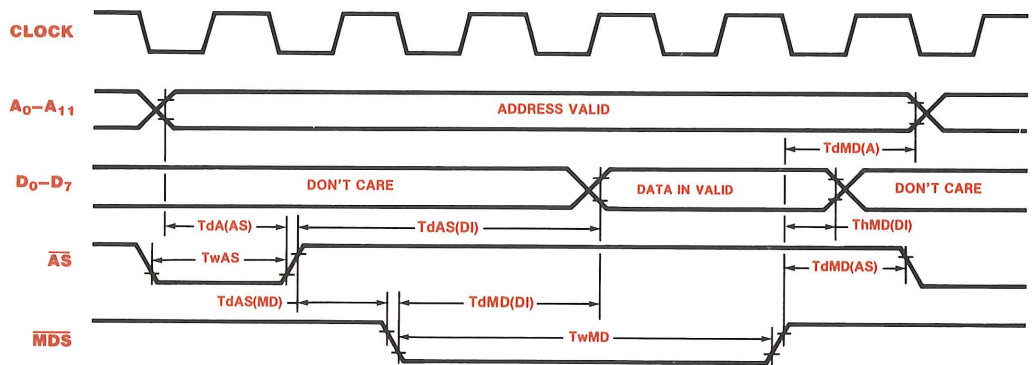


Output Handshake

**Z8/64
Memory Port
Timing**

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 2	1
TdAS(DI)	Address Strobe to Data In Valid Delay Time		280	ns	Test Load 2	3
TwAS	Address Strobe Width	60		ns	Test Load 2	1
TdAS(MD)	Address Strobe to Memory Data Strobe Delay Time	60		ns	Test Load 2	1
TwMD	Memory Data Strobe Width	230		ns	Test Load 2	2
TdMD(DI)	Memory Data Strobe to Data In Valid Delay Time		160	ns	Test Load 2	1
ThMD(DI)	Data In Hold Time	0		ns		
TdMD(A)	Memory Data Strobe to Address Change Delay Time	60		ns	Test Load 2	1
TdMD(AS)	Memory Data Strobe to Address Strobe Delay Time	50		ns	Test Load 2	1

1. Delay times are specified for an input clock frequency of 8 MHz. When operating at a lower frequency, the increase in input clock period must be added to the specified delay time.
2. Data Strobe Width is specified for an input clock frequency of 8 MHz. When operating at a lower frequency, the increase in *three* input clock periods must be added to the specified width. Data Strobe Width varies according to the instruction being executed.
3. Address Strobe and Data Strobe to Data In Valid delay times represent memory system access times and are given for an 8 MHz crystal input frequency. For lower frequencies; the change in four clock periods must be added to TdAS(DI) and the change in three clock periods added to TdDS(DI).
4. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0."



Z8034 UPC Universal Peripheral Controller



**NEW
1980**

Product Brief

Preliminary

February 1980

Features

- Complete slave microcomputer, for distributed-processing Z-Bus use.
- Unmatched power of Z8 architecture, instruction set.
- Three programmable I/O ports, two with 2-wire handshake, or any combination of data and control lines.
- Six levels of priority interrupts to Z-UPC.
- Two programmable 8-bit counter/timers with 6-bit prescalers.
- 256 byte register file, accessible by both master CPU and Z-UPC, as allocated by Z-UPC program.
- 2K bytes of on-chip program ROM for efficiency, versatility.

2

Description

The Z-UPC Universal Peripheral Controller is a distributed microcomputer that performs the three basic interfacing functions needed to interface a CPU with peripherals: device control by ROM-resident internal software, data manipulation, such as reformatting or arithmetic, and data buffering in internal registers.

The Z-UPC is similar to the Z8 microcomputer and uses the Z8 instruction set. Under

program control, its three 8-line I/O ports can be tailored to the needs of its user. Permanently configured as a single-chip controller with 2K bytes of internal ROM, the Z-UPC executes instructions in 2.2 μ s average using a 4-MHz clock source. Its register file contains 256 bytes, of which 234 are general-purpose registers, 19 are status and control registers, and three are port registers.

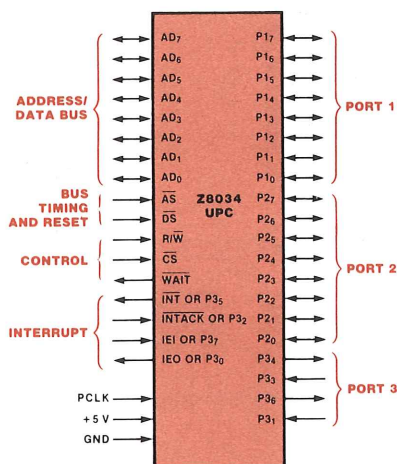


Figure 1. Pin Functions

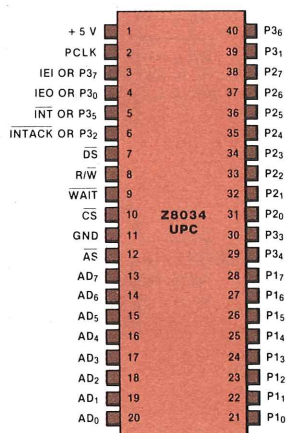


Figure 2. Pin Assignments

Description (Continued)

The Z-UPC Universal Peripheral Controller is an intelligent device that generates all the control signals peripheral devices need. Because it does off-line arithmetic, translates data before transmitting, and buffers data, the Z-UPC unburdens the master CPU, thereby increasing the overall speed and efficiency of the system in which it resides.

Based upon the Z8 microcomputer architecture, the Z-UPC offers fast execution time, efficient use of memory, and sophisticated interrupt, I/O, and bit manipulation. Its powerful and extensive instruction types, combined with its efficient internal register addressing scheme, not only speeds program execution, but also efficiently packs program into the on-chip ROM.

A unique characteristic of the Z-UPC is its register file, which contains I/O port and control registers that can be accessed both by the Z-UPC program and by its associated master CPU. This results in byte efficiency, programming efficiency, and address space efficiency because Z-UPC instructions can operate directly on I/O data without moving it to and from an accumulator. It also allows the Z-UPC user to allocate as data buffer between the CPU and the peripheral all register space not in use as

accumulators, address pointers, index registers, or stack. Registers not used as buffer are protected against CPU access.

The register file is divided into 16 groups of 16 working registers each. A register pointer allows fast, short-format instructions to access any one of these groups quickly, resulting in fast and easy task switching. Two-way communication between the master CPU and the register file is facilitated by another pointer that positions 16 interface registers anywhere within the register file. These registers are accessed directly by both the master CPU and the slave Z-UPC. Four more registers, similarly accessed, convey control and status information.

All of Z-Bus's daisy-chained priority interrupt system can be implemented in the Z-UPC under software control, or the Z-UPC can be programmed to function in a polled environment. In all, the Z-UPC has 24 pins that can be dedicated to I/O functions. Grouped logically into three 8-line ports, they can be programmed in many combinations of inputs, outputs, and bidirectional lines, with or without handshake and with push-pull or open-drain outputs.

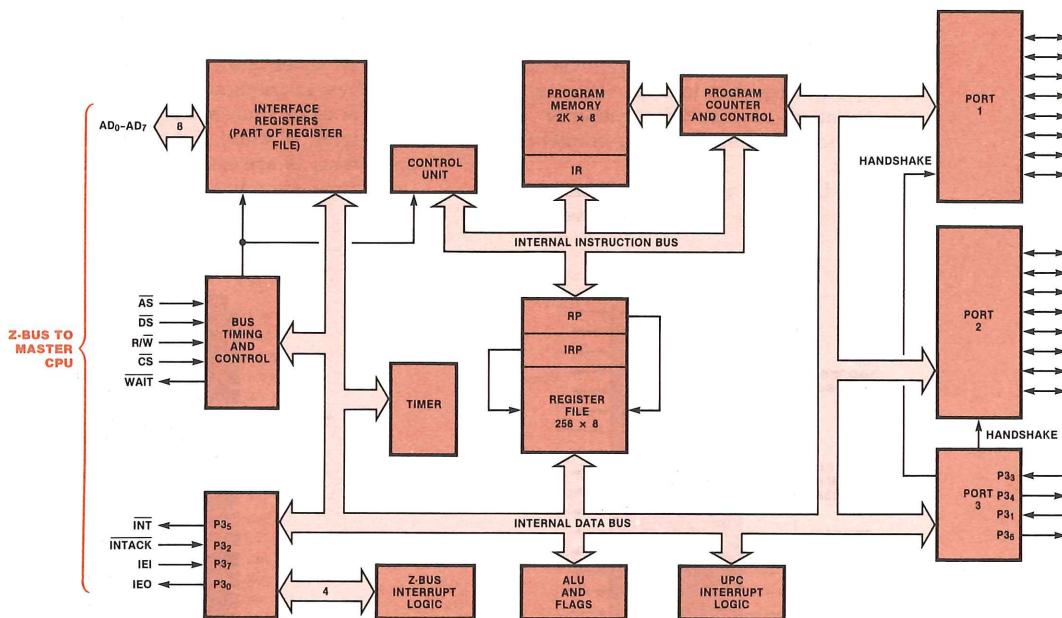


Figure 3. Functional Block Diagram

Z8030 SCC Serial Communications Controller



NEW
1980

Product Brief

Preliminary

February 1980

Features

- Two independent, 0 to 1M bit per second, full-duplex channels, each with its own quartz oscillator, baud-rate generator, and digital phase-locked loop for clock recovery.
- Multi-protocol operation under program control.
- Asynchronous mode with 5 to 8 bits and 1, 1½, or 2 stop bits per character, programmable clock factor, break detection and generation, and parity, overrun, and framing error detection.
- Local loopback and auto-echo modes.
- Bisynchronous mode with internal or external character synchronization on one or two sync characters and CRC generation and checking with CRC-16 or CRC-CCITT preset to either 1s or 0s.
- SDLC/HDLC mode with comprehensive frame-level control, automatic zero insertion and deletion, I-field residue handling, abort generation and detection, CRC generation and checking, and loop mode operation.
- Programmable for NRZ, NRZI, or FM coding.

Description

The Z-SCC Serial Communication Controller is a dual-channel, multi-protocol data communication peripheral for Z-Bus use. It is software-configured to satisfy a wide variety of serial communication applications. Its basic function is serial-to-parallel and parallel-to-serial conversion. However, the Z-SCC also contains a repertoire of new, sophisticated internal functions that minimize the need for

external random logic on the circuit card.

The Z-SCC handles asynchronous formats, synchronous byte-oriented protocols such as IBM Bisync, and synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device also supports virtually any other serial data transfer application (cassette or diskette interface, for example).

The device can generate and check CRC

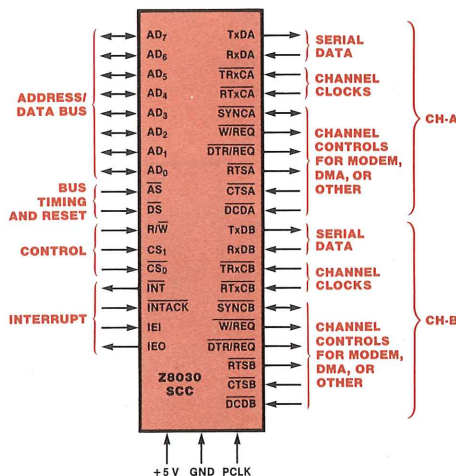


Figure 1. Pin Functions

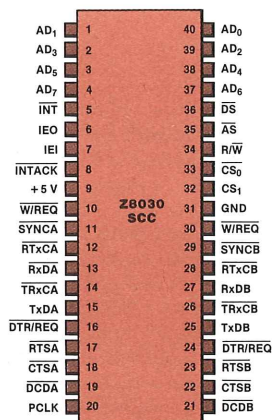


Figure 2. Pin Assignments

Description
(Continued)

codes in any synchronous mode and can be programmed to check data integrity in various modes. It also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

As is standard among Zilog peripheral components, the Z-Bus daisy-chain interrupt hierarchy is supported.

The Z-SCC contains the necessary multiplexed address/data bus interface with strobe and chip select lines to function as a Z-Bus peripheral. It includes internal control and interrupt logic, two full-duplex channels and two baud-rate generators. Associated with each channel are several read and write registers for mode control as well as the logic necessary to interface to modems or other external devices.

The read and write register group for each channel includes eight control registers, two sync-character registers, and four status registers. Each baud rate generator has two read/write registers for holding the time constant that determines baud rate. Associated with the interrupt logic is a write register for interrupt vector and three read registers: vector with status, vector without status, and interrupt pending status.

The logic for both channels provides formatting, synchronization and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general purpose in nature and optionally can be used for functions other than modem control.

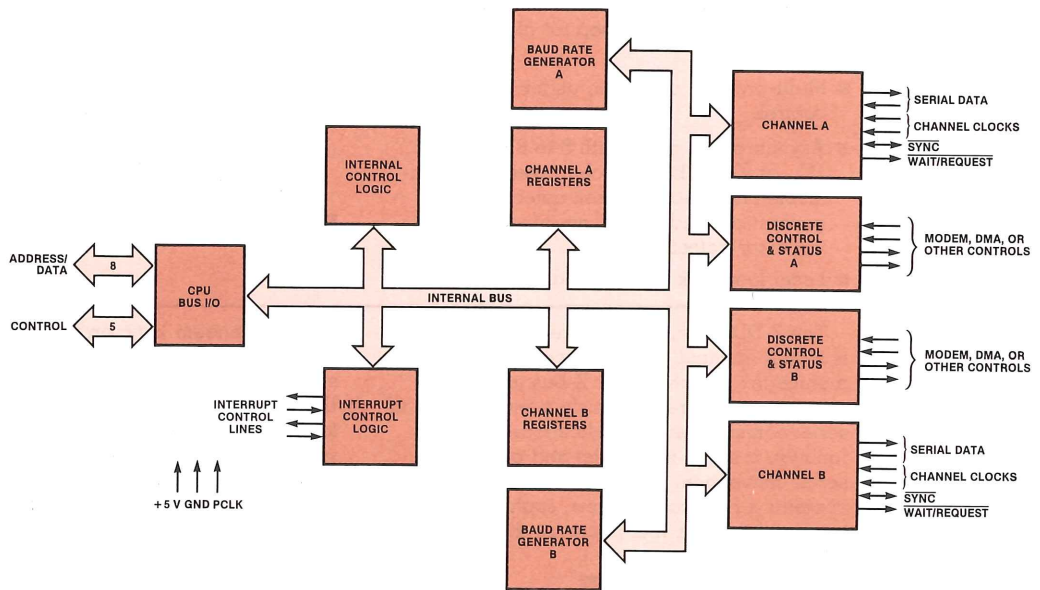


Figure 3. Functional Block Diagram

Typical Applications

Figure 4 shows how a Z-SCC can be connected with channel A programmed for the Synchronous Data Link Control (SDLC) Loop mode, functioning as a secondary station. If NRZI or FM coding is used, no clock lines are required because the clock can be recovered from the received data, using the Z-SCC's on-chip digital phase locked loop (DPLL). Another Z-SCC (not shown), programmed for the SDLC mode, would be the controlling station, polling the loop for traffic. The figure shows a typical, asynchronous serial port being serviced by channel B of the Z-SCC. It could just as well support another synchronous data link, or even a high-speed link, transferring data via a DMA controller.

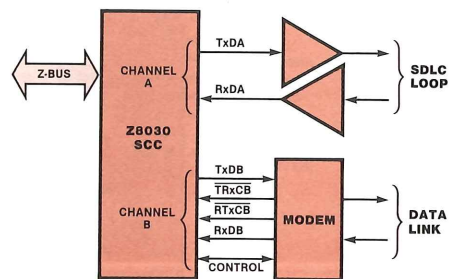


Figure 4. Loop Secondary Station and Serial Port

Z8036 CIO Counter/Timer and Parallel I/O Unit



NEW
1980

Product Brief

Preliminary

February 1980

Features

- Two independent 8-bit double-buffered bidirectional I/O ports plus a special-purpose 4-bit I/O port.
- Four handshake modes including IEEE-488.
- Wait/Request line for high speed data transfer.
- Three independent 16-bit counters.
- All registers read/write and directly addressable.
- Flexible pattern recognition logic, programmable as 16-input interrupt controller.

2

Description

The Z8036 CIO Counter/timer and Parallel I/O element is a general-purpose peripheral circuit that satisfies most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications. The use of the device is simplified by making all internal registers (command, status, and data) readable and (except for status bits) writable. Also, each register is given its own unique address so it can be accessed directly—no special sequential operations are required. The Z-CIO is directly Z-Bus compatible.

Either 8-bit I/O port can be a handshake

byte port or a bit port. In the bit mode, data direction is programmable bit by bit. In the handshake mode, the ports can be input, output, or bidirectional, and they may be linked to form a 16-bit port. The four handshake modes include IEEE-488, interlocked (for interfacing to a Z-UPC, Z-FIO or another Z-CIO), strobed and pulsed. The pulsed mode connects one counter/timer with the handshake logic for interfacing a mechanical device such as a printer. The 4-bit port provides handshake controls, special controls (Wait/Request) or general-purpose I/O.

The counter/timer section contains three 16-bit counters, two of which can be software-configured as a 32-bit counter/timer. Up to

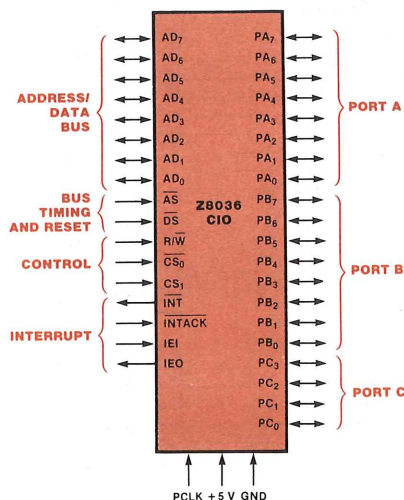


Figure 1. Pin Functions

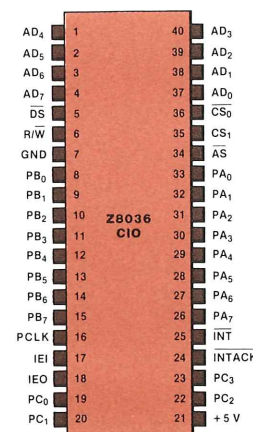


Figure 2. Pin Assignments

Description
(Continued)

four I/O lines for each counter are available for direct external control and status information. All counters have a programmable output duty cycle, continuous or single-cycle operation, and the counting process can be programmed to be either retrigged or nonretrigged.

Figure 3 shows how the Z-CIO is used. The two general purpose 8-bit ports are similar. They can be programmed as handshake driven, double-buffered ports (input, output, or bidirectional) or as control ports in which the direction of each bit is individually programmable. Port B can also be specified to provide external access for two of the counter/timers. Each port includes pattern recognition logic allowing interrupt generation when a specified pattern is detected. The pattern recognition logic can be programmed so that the port functions like a priority interrupt controller.

To control these capabilities, each port contains 13 registers. Three of these, the input, output, and buffer registers, are data path registers. Two others, the mode specification and handshake specification registers, define the mode of the port and specify what handshake to use, if any. The reference pattern for the pattern recognition logic is defined in three registers, the pattern polarity, pattern transition, and pattern mask registers. The detailed characteristics of each bit path (for example, the direction of data flow, or whether a path is inverting or noninverting) are programmed using the data path polarity, data direction, and special I/O control registers. The primary control and status bits are grouped in a single register so that after the ports are configured initially, only this register

need be accessed often. One register contains the interrupt vector associated with each port. To facilitate initialization, the port logic is designed so that if a capability of the port is not required the registers associated with that capability are ignored and need not be programmed.

The function of port C depends upon the roles of ports A and B. Port C provides handshake lines for the other two when required. Any bits of port C not so used can be used as I/O lines or as external access to the third counter/timer.

Besides the data input and output registers, three registers are needed. These specify the details of each bit path: data path polarity, data direction, and special I/O control.

The three counter/timers are all identical. Each is composed of a 16-bit down-counter, a 16-bit time constant register (which holds the value loaded into the down-counter), a 16-bit current count register (used to read the contents of the down-counter), and two 8-bit registers for control and status (the mode select and control registers). All three share a common vector register.

Each counter/timer can be programmed as either counter or timer. Up to four port I/O lines can be designated as external access lines for it. The lines are: Counter Input, Gate Input, Trigger Input, and Counter/Timer Output. Three different counter/timer output duty cycles are available: pulse, one-shot, or square wave. The operation of the counter/timer can be specified to be either single cycle or continuous. The counting sequence may be retrigged or nonretrigged, under program control.

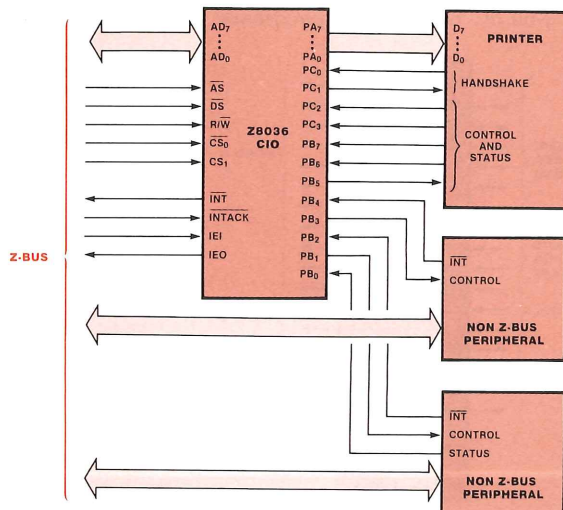


Figure 3. Functional Block Diagram

Z8038 FIO FIFO Input/Output Interface Unit



**NEW
1980**

Product Brief

Preliminary

February 1980

Features

- Asynchronous bidirectional FIFO buffer, used with most major microprocessors as CPU/CPU or CPU/peripheral interface.
- Interlocked or IEEE-488 handshake port mode; Empty, Full, and Wait/Request lines for high-speed data transfer.
- 128 x 8 organization, expandable to any width; cascable to any depth.

- Preset byte count in FIO buffer can interrupt CPU.
- All registers directly addressable.
- Vectored/non-vectored interrupts on pattern/status match, over/underflow error, buffer status.

2

Description

The Z-FIO is a general-purpose microprocessor interface that provides elastic buffering between asynchronous CPUs in a parallel-processor network or between CPU and peripheral circuits. The Z-FIO can interface a Z-Bus microprocessor or any other major processor to another microprocessor or to a peripheral circuit or port.

In Z8000 systems, the FIO furthers distributed-processor operation because it can interconnect components or subsystems operating at different speeds. Also, it can increase system throughput by transferring

words as well as bytes. This bidirectional device accepts data and holds it until it can be used by another device in the system. In most I/O transactions, introducing a 128-deep buffer cuts interrupt servicing overhead by two orders of magnitude.

The Z-FIO greatly facilitates system throughput by moving variable-size blocks under either direct memory access or interrupt control—an especially important consideration when fast peripheral circuits need interfacing. Complete status information is also provided for operation in polled environments.

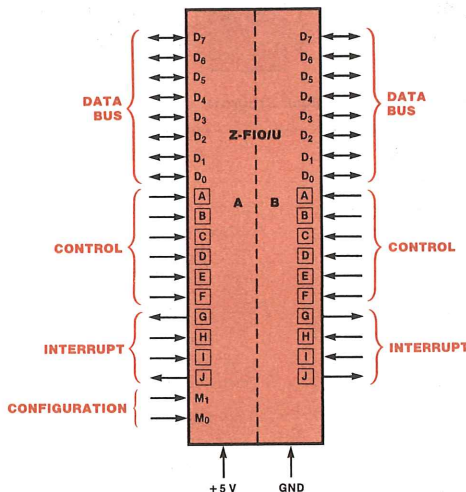


Figure 1. Pin Functions

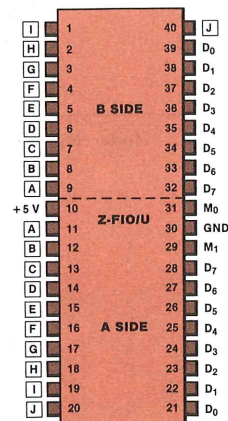


Figure 2. Pin Assignments

Pin Assignments	Z-Bus Low Byte	Z-Bus High Byte	General Purpose	Interlocked HS Port*	IEEE 488 HS Port*
A	REQ/WT	REQ/WT	REQ/WT	RFD/DAV	RFD/DAV
B	DMASTB	DMASTB	DACK	ACKIN	DAV/DAC
C	DS	DS	RD	FULL	DAC/RFD
D	R/W	R/W	WR	EMPTY	EMPTY
E	CS	CS	CS	CLEAR	CLEAR
F	AS	AS	C/D	DATA DIR	DATA DIR
G	INTACK	A ₀	INTACK	IN ₀	IN ₀
H	IEO	A ₁	IEO	OUT ₁	OUT ₁
I	IEI	A ₂	IEI	IN ₂	IN ₂
J	INT	A ₃	INT	OUT ₃	OUT ₃

*B side only. See table below.

Description (Continued)

The internal functions of the Z-FIO are shown in the block diagram (Figure 3). It is made up of two sides that are identical except for programming. The side programmed by pins M₀ and M₁ is called the A side; the side programmed by bits SL₀ and SL₁ is called the B side. Common to both, and situated between the two sides, is the 128 × 8 RAM used for data storage, two 7-bit counters, and various registers. The RAM is capable of simultaneous, independent read and write operations. This means, for example, that the A side CPU can write a byte of data into the FIO without disturbing a simultaneous read operation by the B side CPU. The outputs of the read and write counters are used to address the buffer RAM,

and also are fed into a subtractor to determine the current number of bytes in the memory. This number can be read by either CPU from a status register dedicated to each side. Another programmable register is compared against the status register to generate an interrupt and/or start and stop DMA transfers. A pair of port registers allows for communication between CPUs, bypassing the main buffer memory.

Operating Modes. The Z-FIO has twelve different programmable modes. (Table below.) The states of two package pins determine the mode of operation of the A side, and the B side is programmed by two bits (SL₀ and SL₁) in one of the A side control registers.

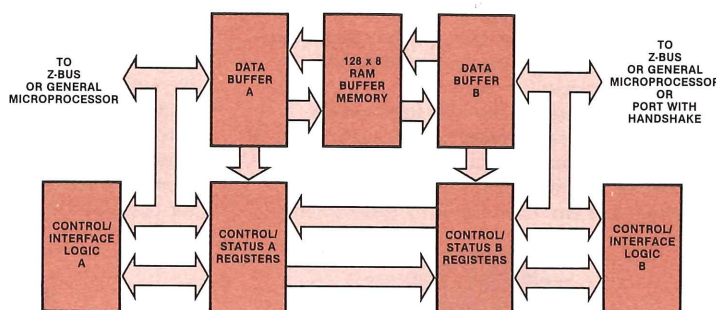


Figure 3. Functional Block Diagram

Operating Modes	Mode	M ₁	M ₀	SL ₁	SL ₀	A	B
	0	0	0	0	0	Z-Bus Low Byte	Z-Bus Low Byte
	1	0	0	0	1	Z-Bus Low Byte	General μP
	2	0	0	1	0	Z-Bus Low Byte	IEEE 488 HS
	3	0	0	1	1	Z-Bus Low Byte	Interlocked HS
	4	0	1	0	0	Z-Bus High Byte	Z-Bus High Byte
	5	0	1	0	1	Z-Bus High Byte	General μP
	6	0	1	1	0	Z-Bus High Byte	IEEE 488 HS
	7	0	1	1	1	Z-Bus High Byte	Interlocked HS
	8	1	0	0	0	General μP	Z-Bus Low Byte
	9	1	0	0	1	General μP	General μP
	10	1	0	1	0	General μP	IEEE 488 HS
	11	1	0	1	1	General μP	Interlocked HS

Z8060 FIFO FIFO Buffer Unit and Z-FIO Expander



NEW
1980

Product Brief

Preliminary

February 1980

Features

- Asynchronous, bidirectional first-in, first-out buffer.
- Extends depth of Z-FIO without limit.
- 128 x 8 organization.
- 3-state data outputs.
- Empty and Full status pins are wire-ORed among multiple stages.

Description

The Z-FIFO first-in, first-out buffer unit is a 128 x 8-bit memory with bidirectional data transfer capability and handshake logic. Its structure is similar to that of other FIFOs that are commonly available, such as the AM2812 and the 3351. The handshake logic used is compatible with that of the Z8, the Z-CIO, and Z-FIO. Z-FIFO buffers can be cascaded, end to end, without limit, their RFD/DAV and ACKIN signals daisy-chained, to make a FIFO array any desired number of words deep. Two such channels in parallel, suitably controlled, make up a 16-bit-wide buffer array.

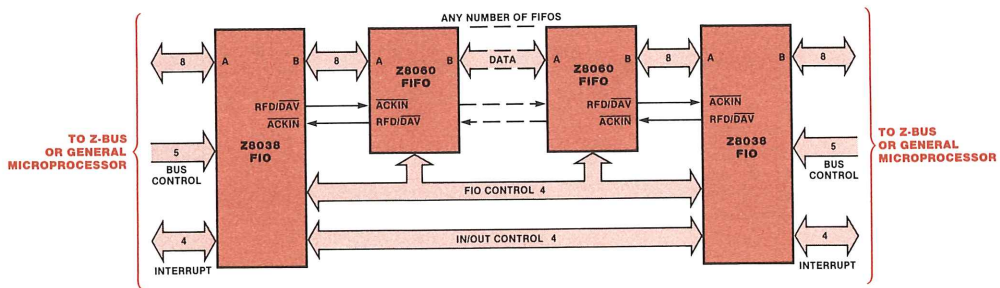
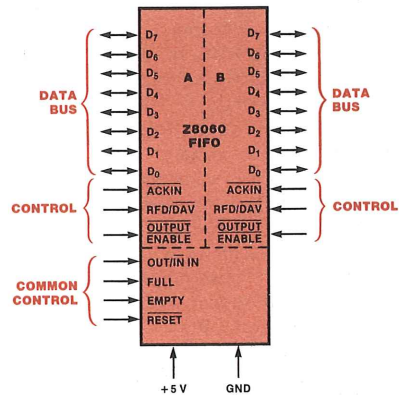


Figure 2. Using FIFOs to Extend FIOs

More to Come

The components described in the foregoing Product Briefs exist now or are well along in their development. These components represent the first step of an ongoing commitment to support the Z8000 Family. The next step is a series of specialized processors and peripherals that includes the Z8016 DMA Controller, Z8052 CRT Controller and Z8065 Burst Error Processor.

Z8016 DMA Controller. This high-speed (2M byte/sec) versatile dual-channel DMA controller matches the power and addressing capability of the Z8000 CPUs. It supports a variety of system implementations, ranging from dedicated single-DMA configurations to distributed multiple-DMA configurations found in multi-user, multi-tasking environments.

The Z8016 takes full advantage of the Z8000 memory management scheme because it interfaces directly to the Z8010 Memory Management Unit. Consequently, 8M bytes of logical address range are provided for each CPU address space. Alternatively, the Z8016 can operate independently of the Z8010 MMU and directly address up to 16M bytes of physical address space.

The ability to self-load control parameters from memory enables chained DMA operations of different types, and provides a high degree of independence from the CPU. Memory-to-Memory data transfers as well as the more conventional data transfer between I/O and memory can be executed. Data transfers can be in the form of single byte, double byte or word, and a number of search and match operations can be performed on the data.

Several different interrupt stimuli can be enabled under program control. These include interrupts on Terminal Count (TC), End of Process (EOP), or a number or condition bits in the Channel Mode Register. The Z8016 operates within the Z8000 daisy-chain vectored-priority interrupt scheme.

Z8052 CRT Controller. The Z8052 is a broad-application, raster scan CRT controller specifically designed to interface the Z8000 CPU to a variety of CRT displays.

Its numerous advanced features suit a wide variety of applications including general-business and scientific data processing, word processing and graphics.

Text-editing software implementation is simplified by means of variable vertical and horizontal split-screen capability. Double character-cell display with vertical adjustment of character position enhances word processing as well as scientific and mathematical notation. Character justification along with mixing of fonts is achieved by external synchronization of vertical retrace and variable character clock frequency. Typewriter formatted single- or multiple-line spacing along with oversized alpha-numerics or memory bit-mapped displays can be generated with variable scan lines per character row. Simple line drawing capability is also made available with the line attributes provided.

The Z8052 programmable register architecture allows easy operating configuration and mode changes under software program control. This contrasts favorably to the time consuming and costly ROM programming methods used by other CRT controllers.

These advanced display features are achieved with the ability to change attributes dynamically on a real-time basis. The ability to change row attributes in real time allows smooth scrolling. Character attributes changing in real time allow multiple cursors and various underline combinations, all with individual blink rates. The character attributes control normal or double character cells for subscripted and superscripted display as well as the line attributes used for line drawing. This allows intermixing of forms and text displays for various applications.

Dynamic change of attributes is achieved by using a separate display refresh RAM in which the data for changing attributes is stored in parallel with the displayed character data. Data transfers from host processor to display RAM are made without memory access contention problems between Z8052 CRTC and processor. This is achieved by setting up the display RAM addressing so the first half of the character clock period is used for CRTC functions and the second half is used for host processor data transfer.

Z8065 Burst Error Processor.

The Z8065 Burst Error Processor provides high-speed error correction for Z8000 disk-based systems or other Z8000 systems in which high-speed data transfer takes place. With data transfer rates and storage size increasing in both floppy and solid disks, error rates without correction become increasingly intolerable. The Z8065 complements the Z8000 and its peripherals in systems using high-speed or high-density peripheral interfaces such as solid disk or floppy disk drives.

The Z8065 provides a selection of the four generator polynomials popularized by IBM, Digital Equipment Corporation and Calcomp. The divide mode performs division of the data stream by the selected polynomial to generate the check bits during data transfer to disk. For read operations, the divide mode can detect errors that may have occurred in a read or write. To locate the error for correction, a second compute mode identifies the error pattern and assists in finding the error location in the data stream.

3

Memories
Zilog

Z6115 8K x 1 Dynamic RAM



NEW
1980

Product Specification

Preliminary

February 1980

Features

- 8,192 words by 1-bit organization
- Industry-standard 16-pin package
- $\pm 5\%$ tolerance on all power supplies
- Low power consumption: 462 mW active; 20mW standby (max)
- Column Address $A_0=0$ for upper half matrix (Z6115A). Column Address $A_0=1$ for lower half matrix (Z6115B)
- On-chip address and data-in latches

- Read/modify/write, page-mode and $\overline{\text{RAS}}$ -only refresh capabilities
- Output data controlled by $\overline{\text{CAS}}$ and unlatched at end of cycle
- Common I/O capability using the early-write mode
- 1 ms refresh period, 128-cycle refresh

	Z6115
Maximum Access Time (ns)	250
Read, Write Cycle (ns)	410
Read/Modify/Write Cycle (ns)	500

General Description

The Z6115 is a 8,192-bit, high-speed dynamic random-access memory designed for high-performance, low-cost applications in mainframe and buffer memories and peripheral storage. Organized as 8,192 1-bit words and fabricated using a double-poly n-channel silicon-gate process, this device optimizes speed, power and density tradeoffs.

By multiplexing row and column address inputs, the Z6115 requires only seven address lines and permits packaging in a standard 16-pin dual in-line package. This packaging technique allows high system density and is compatible with widely available automated test and insertion equipment. Complete address decoding is done on chip.

All inputs are TTL compatible, and the output is 3-state TTL compatible. On-chip address

and data registers eliminate the need for interface registers.

The single transistor storage cell requires refreshing for data retention. Refreshing is accomplished by performing $\overline{\text{RAS}}$ -only refresh cycles, hidden refresh cycles, or normal read or write cycles on the 128 address combinations of A_0 through A_6 during a 1 ms period. A write cycle refreshes stored data on all bits of the selected row except the bit which is addressed.

In addition to the usual read, write and read/modify/write cycles, the Z6115 is capable of delayed write cycles, page-mode operation and $\overline{\text{RAS}}$ -only refresh. Furthermore, proper control of $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and $\overline{\text{WRITE}}$ permits common I/O capability, two-dimensional chip selection and extended page boundaries.

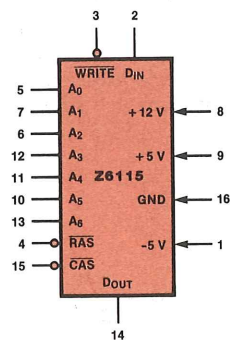


Figure 1. Logic Symbol

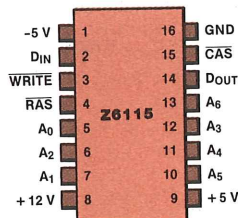


Figure 2. Pin Assignments

A_0-A_6	Address Inputs
DIN	Data Input
DOUT	Data Output
CAS	Column Address Strobe
RAS	Row Address Strobe
WRITE	Read/Write Input
-5 V	-5 V Power Supply (V_{BB})
+5 V	+5 V Power Supply (V_{CC})
+12 V	+12 V Power Supply (V_{DD})
GND	Ground (V_{SS})

Pin Names

Addressing. The 13 address bits which are required to decode one of the 8,192 cell locations within the Z6115 are multiplexed onto the seven address inputs and latched into the on-chip address latches by externally applying two negative-going TTL-level clocks. The first clock, Row Address Strobe ($\overline{RAS}$), latches the seven row address bits into the chip. The second clock, Column Address Strobe ($\overline{CAS}$), triggers a sequence of events that are controlled by different delayed internal clocks. $\overline{CAS}$ subsequently latches the six column addresses (A_1 - A_6) into the chip. At Column Address Strobe time, the logical state of A_0 (pin 5) must be at the level specified by the Z6115A or B. If a Z6115A is utilized, A_0 must be at a logic 0 (0.8 V max.). If a Z6115B is utilized, A_0 must be at a logic 1 (2.4 V min.). The other six address bits used for column addresses (A_1 - A_6) function normally. The two clock chains are linked together logically so that the address multiplexing operation is done outside the critical-path timing sequence for read data access. The later events in the $\overline{CAS}$ clock sequence are inhibited until a delayed signal derived from the $\overline{RAS}$ clock chain appears.

This "gated- $\overline{CAS}$ " feature allows the $\overline{CAS}$ clock to be externally activated as soon as the Row Address Hold Time specification has been satisfied and the seven address inputs have been changed from row address to column address information.

$\overline{CAS}$ can be activated any time after t_{RAH} with no effect on the worst-case data access time (t_{RAC}), up to the point when the delayed row clock no longer inhibits the remaining sequence of column clocks.

Two timing end points result from the internal gating of $\overline{CAS}$ and are called $t_{RCD(min)}$ and $t_{RCD(max)}$. No data or storage errors result if $\overline{CAS}$ is applied to the Z6115 any time beyond the $t_{RCD(max)}$ limit. However, access time is then determined exclusively by the access time from $\overline{CAS}$ (t_{RAC}), and the access time from $\overline{RAS}$ is lengthened by the amount that t_{RCD} exceeds the $t_{RCD(max)}$ limit.

Data Input/Output. An on-chip register latches information on the Data In pin whenever $\overline{RAS}$ is active and both $\overline{CAS}$ and $\overline{WRITE}$ have made their negative transitions. The later of the two signals ($\overline{CAS}$ or $\overline{WRITE}$) that makes its negative transition generates the

strobe for the Data In register. This feature is illustrated by referencing D_{IN} to $\overline{WRITE}$ in the timing diagrams that depict the read/write and page-mode write cycles, while the early-write cycle diagram shows D_{IN} referenced to $\overline{CAS}$.

In the read cycle, data is retrieved from the memory by maintaining $\overline{WRITE}$ inactive throughout the portion of the memory cycle in which $\overline{CAS}$ is active. Data then appears at the output within the specified access time.

Unlike most 16-pin 4K dynamic RAMs and some 16K dynamic RAMs, the Data Output pin of the Z6115 is normally in the high-impedance state ($\overline{CAS}$ inactive). The only time the output buffer turns on and presents a logic 0 or 1 at the D_{OUT} pin is at access time during a read, read/modify/write or delayed-write cycle. The output remains valid from access time until $\overline{CAS}$ returns High. The output data is not inverted (that is, it has the same polarity as the input data).

In the early-write cycle, however, the output pin stays in the high-impedance state throughout the entire cycle. This type of $\overline{CAS}$ -controlled latch provides features that are important in the following system applications:

- **Common I/O Operation.** If all write operations are performed in the early-write mode, then D_{IN} and D_{OUT} can be tied together for a common I/O data bus.
- **Flexible $\overline{RAS}/\overline{CAS}$ Clock Timing.** D_{OUT} is controlled only by $\overline{CAS}$. It remains valid during a read cycle from access time until $\overline{CAS}$ returns High. The data can be maintained valid from one cycle until a new memory cycle begins with no penalty in cycle time. Thus, $\overline{CAS}$ and $\overline{RAS}$ can return to their inactive states independent of each other, provided the t_{RAS} and t_{CAS} specifications are met.
- **Two Methods of Chip Selection.** Since $\overline{CAS}$ is not required to turn off the outputs of unselected memory devices in a matrix (there is no special output deselection cycle), both $\overline{CAS}$ and/or $\overline{RAS}$ can be decoded for chip selection. If both $\overline{CAS}$ and $\overline{RAS}$ are decoded, a two-dimensional (X, Y) chip-selected array can be constructed.
- **Extended Page Boundary.** If $\overline{CAS}$ is decoded as a page-select signal, the page boundary can be extended beyond the 128 column locations in a single chip.

Functional Description
(Continued)

Page Mode Operation. Page mode operation allows successive memory cycles at multiple column locations of the same row address. This mode is achieved by latching the row address into the chip and maintaining $\overline{\text{RAS}}$ active throughout all successive memory cycles in which the row address is unchanged. This mode of operation results in power savings (power associated with the negative-going edge of $\overline{\text{RAS}}$ is not dissipated), and improved access and cycle times. (Time required for strobing in a row address is eliminated.)

The page boundary of a single Z6115 is limited to the 128 column locations determined by all combinations of the seven address bits. However, in system applications that require more than 8K words, the page boundary can be extended by using $\overline{\text{CAS}}$ rather than $\overline{\text{RAS}}$ as

the chip-select signal. $\overline{\text{RAS}}$ is applied to all devices to latch the row address into each device. $\overline{\text{CAS}}$ is then decoded and serves as a page-cycle select signal. Only those devices that receive both $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ execute a read or write cycle.

Refresh. The dynamic cell matrix is refreshed with a memory cycle at each of the 128 row addresses within each 1 ms time interval. Although any normal memory cycle accomplishes refresh, this function is best done with $\overline{\text{RAS}}$ -only cycles, because $\overline{\text{RAS}}$ -only refresh results in a substantial reduction in operating power. During standby operation, V_{CC} power may be removed without affecting refresh operation.

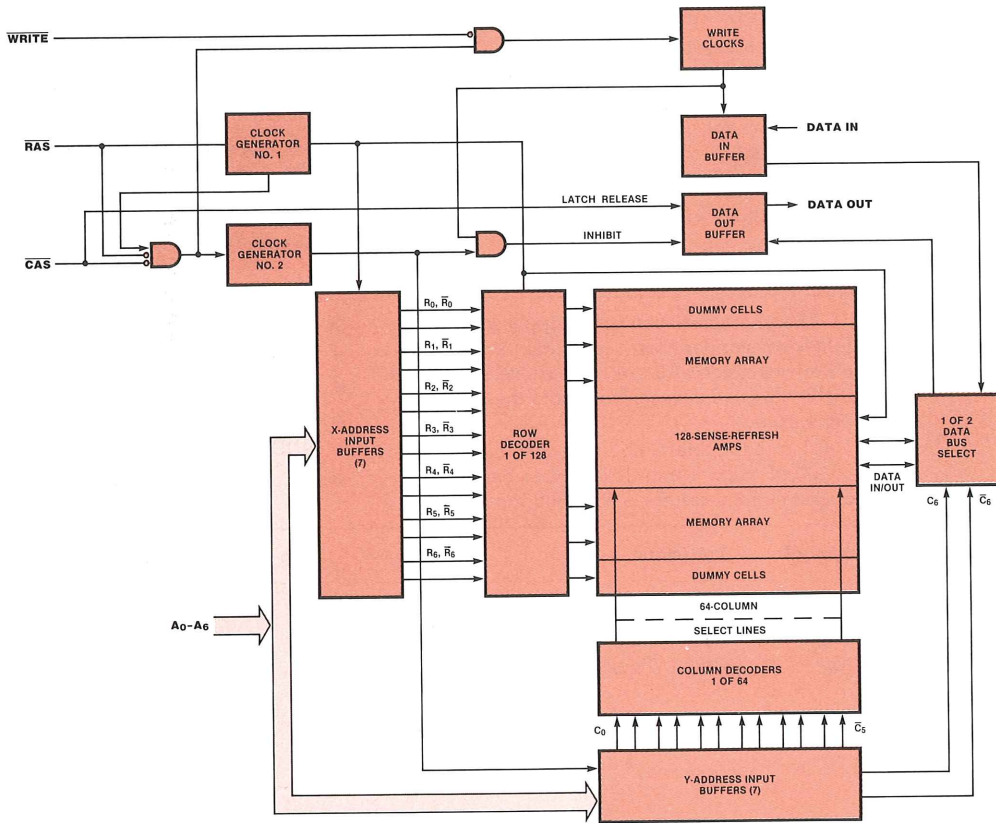


Figure 3. Block Diagram

Power Considerations

Because most of the circuitry on the Z6115 is dynamic, power dissipation is primarily a function of operating frequency rather than duty cycle. In applications requiring lower power dissipation, the device can be operated at a lower frequency (longer cycle time) to reduce the average power dissipation.

Although $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ can be decoded and used as a chip select signal for the Z6115, overall system power is minimized if only $\overline{\text{RAS}}$ is used for this purpose. All unselected devices (those that do not receive $\overline{\text{RAS}}$) remain in a low-power (standby) mode, regardless of the state of $\overline{\text{CAS}}$.

Although no particular power supply noise restrictions exist, other than the supply voltages remaining within the specified tolerance limits, adequate decoupling should be provided to suppress high-frequency noise

resulting from the transient current of the device. This insures optimum system performance and reliability. Bulk capacitance requirements are minimal since the Z6115 draws very little dc current.

The Z6115 requires no special power supply sequencing as long as the Absolute Maximum Rating conditions are observed. To insure compliance with the Absolute Maximum Ratings, Zilog recommends that -5 V be applied first and removed last. The -5 V connection must never be more positive than GND when power is applied to the +12 V connection.

After power is applied to the device, the Z6115 requires several cycles before proper operation of the device is achieved. Any eight cycles that perform refresh satisfy this requirement.

Absolute Maximum Ratings

Voltage on any pin relative to V_{BB} -0.5 V to +20 V
Voltage on V_{DD} , V_{CC} supplies relative to V_{SS} -1.0 to +15.0 V
 $V_{\text{BB}} - V_{\text{SS}}$ ($V_{\text{DD}} - V_{\text{SS}} > 0$ V) 0 V
Ambient operating temperature, T_{A} 0°C to +50°C
Ambient storage temperature -65°C to +150°C

Short circuit output current 50 mA
Power dissipation 1 Watt

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

All ac parameters assume a load capacitance of 100 pF max. Timing references between two output signals assume a load difference of

50 pF max. Standard conditions are as follows:

- $+4.75 \text{ V} \leq V_{\text{CC}} \leq +5.25 \text{ V}$
- $+11.4 \text{ V} \leq V_{\text{DD}} \leq +12.6 \text{ V}$
- $-5.25 \text{ V} \leq V_{\text{BB}} \leq -4.75 \text{ V}$
- $\text{GND} = 0 \text{ V}$
- $0^\circ\text{C} \leq T_{\text{A1}} \leq +50^\circ\text{C}$

Capacitance	Symbol	Parameter	Typ	Max	Units	Notes
C_{11}		Input Capacitance (A_0 - A_6), D_{IN}	4	5	pF	17
C_{12}		Input Capacitance $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, $\overline{\text{WRITE}}$	8	10	pF	17
C_0		Output Capacitance (D_{OUT})	5	7	pF	17, 18

DC Character-istics

Symbol	Parameter	Min	Max	Unit	Notes
V_{DD}	Supply Voltage	11.40	12.60	V	2
V_{CC}	Supply Voltage	4.75	5.25	V	2, 3
V_{SS}	Supply Voltage	0	0	V	2
V_{BB}	Supply Voltage	-4.75	-5.25	V	2
V_{IH}	Input High (Logic 1) Voltage all inputs	2.40	7.00	V	2
V_{IL}	Input Low (Logic 0) Voltage all inputs	-1.00	0.80	V	2
I_{DD1}	OPERATING CURRENT		35.00	mA	4
I_{CC1}	Average power supply operating current				5
I_{BB1}	(RAS, CAS cycling; t_{RC} = min. spec.)		200.00	μ A	
I_{DD2}	STANDBY CURRENT		1.50	mA	
I_{CC2}	Power supply standby current	-10.00	10.00	μ A	
I_{BB2}	(RAS = V_{IH} , D_{OUT} = High impedance)		100.00	μ A	
I_{DD3}	REFRESH CURRENT		27.00	mA	4
I_{CC3}	Average power supply current, refresh mode	-10.00	10.00	μ A	
I_{BB3}	(RAS cycling, $\overline{CAS}$ = V_{IH} t_{RC} = min. spec.)		200.00	μ A	
I_{DD4}	PAGE MODE CURRENT		27.00	mA	4
I_{CC4}	Average power supply current,				5
I_{BB4}	page-mode operation (RAS = V_{IH} , CAS cycling; t_{PC} = min. spec.)		200.00	μ A	
I_{IL}	INPUT LEAKAGE Input leakage current, any input (V_{BB} -5 V, 0 V $\leq V_{IN} \leq$ +7.0 V, all other pins not under test = 0 V)	-10.00	10.00	μ A	
V_{OL}	OUTPUT LEAKAGE Output leakage current (D_{OUT} is disabled, 0 V $\leq V_{OUT} \leq$ +5.5 V)	-10.00	10.00	μ A	
V_{OH}	OUTPUT LEVELS Output high (Logic 1) voltage (I_{OUT} = -5 mA)	2.40		V	3
V_{OL}	Output low (Logic 0) voltage (I_{OUT} = 4.2 ma)		0.40	V	

NOTES:

1. T_A is specified for operation at frequencies to $t_{RC} \geq t_{RC}$ (min). Operation at higher cycle rates with reduced ambient temperatures and higher power dissipation is permissible, provided ac operating parameters are met.
2. All voltages referenced to GND.
3. Output voltage will swing from GND to V_{CC} when activated with no current loading. For purposes of maintaining data in standby mode, V_{CC} may be

reduced to V_{SS} without affecting refresh operations or data retention. However, the V_{OH} (min) specification is not guaranteed in this mode.

4. I_{DD1} , I_{DD3} and I_{DD4} depend on cycle rate.
5. I_{CC1} and I_{CC4} depend upon output loading. During readout of high level data V_{CC} is connected through a low impedance (125 Ω typ) to data out. At all other times I_{CC} consists of leakage currents only.

AC Character-istics	Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes
	1	t_{RC}	Random Read or Write Cycle Time	410		9
	2	t_{RMW}	Read/Modify/Write Cycle Time	500		9
	3	t_{PC}	Page Mode Cycle Time	275		
	4	t_{RAC}	Access Time from $\overline{RAS}$		250	10, 12
	5	t_{CAC}	Access Time from $\overline{CAS}$		165	11, 12
	6	t_{OFF}	Output Buffer Turn-Off Delay	0	60	13
	7	t_T	Transition Time (rise and fall)	3	50	8
	8	t_{RP}	$\overline{RAS}$ Precharge Time	150		
	9	t_{RAS}	$\overline{RAS}$ Pulse Width	250	10K	
	10	t_{RSH}	$\overline{RAS}$ Hold Time	165		
	11	t_{CSH}	$\overline{CAS}$ Hold Time	250		
	12	t_{CAS}	$\overline{CAS}$ Pulse Width	165		
	13	t_{RCD}	$\overline{RAS}$ to $\overline{CAS}$ Delay Time	35	85	14
	14	t_{CRP}	$\overline{CAS}$ to $\overline{RAS}$ Precharge Time	-20		
	15	t_{ASR}	Row Address Set-Up	0		
	16	t_{RAH}	Row Address Hold Time	35		
	17	t_{ASC}	Column Address Set-Up Time	-10		
	18	t_{CAH}	Column Address Hold Time	75		
	19	t_{RCS}	Read Command Set-Up Time	0		
	20	t_{RCH}	Read Command Hold Time	0		
	21	t_{WCH}	Write Command Hold Time	75		
	22	t_{WP}	Write Command Pulse Width	75		
	23	t_{RWL}	Write Command to $\overline{RAS}$ Lead Time	85		
	24	t_{CWL}	Write Command to $\overline{CAS}$ Lead Time	85		
	25	t_{DS}	Data In Set-Up Time	0		15
	26	t_{DH}	Data In Hold Time	75		15
	27	t_{CP}	$\overline{CAS}$ Precharge (page mode only)	100		
	28	t_{REF}	Refresh Period		1 ms	
	29	t_{WCS}	WRITE Command Set-Up Time	-20		16
	30	t_{CWD}	$\overline{CAS}$ to WRITE Delay	90		16
	31	t_{RWD}	$\overline{RAS}$ to WRITE Delay	175		16

6. Several cycles are required after power-up before proper device operation is achieved. Any 8 cycles which perform refresh are adequate for this purpose.

7. AC measurements assume $t_T = 5$ ns.

8. V_{IH} (min) and V_{IL} (max) are reference levels for measuring timing of input signals. Also, transition times are measured between V_{IH} and V_{IL} .

9. The specifications for t_{RC} (min) and t_{RWC} (min) are used only to indicate cycle time at which proper operation over the full temperature range ($0^\circ\text{C} \leq T_A \leq 70^\circ\text{C}$) is assured.

10. Assumes that $t_{RCD} \leq t_{RCD}(\text{max})$. If t_{RCD} is greater than the maximum recommended value shown in this table, t_{RAC} increases by the amount that t_{RCD} exceeds the value shown.

11. Assumes that $t_{RCD} \geq t_{RCD}(\text{max})$.

12. Measured with a load equivalent to 2 TTL loads and 100 pF.

13. $t_{OFF}(\text{max})$ defines the time at which the output achieves the open circuit condition and is not referenced to output voltage levels.

14. Operation within the $t_{RCD}(\text{max})$ limit insures that $t_{RAC}(\text{max})$ can be met. $t_{RCD}(\text{max})$ is specified as a reference point only; if t_{RCD} is greater than the specified $t_{RCD}(\text{max})$ limit, then access time is controlled exclusively by t_{CAC} .

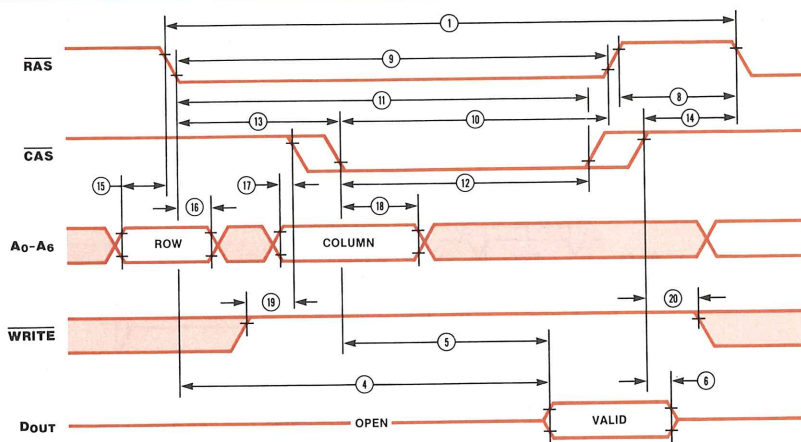
15. These parameters are referenced to $\overline{CAS}$ leading edge in early-write cycles and to WRITE leading edge in delayed write or read/modify/write cycles.

16. t_{WCS} , t_{CWD} , and t_{RWD} are not restrictive operating parameters. They are included in the data sheet as electrical characteristics only: if $t_{WCS} \geq t_{WCS}(\text{min})$, the cycle is an early-write cycle and the data out pin will remain open circuit (high impedance) throughout the entire cycle. If $t_{CWD} \geq t_{CWD}(\text{min})$ and $t_{RWD} \geq t_{RWD}(\text{min})$, the cycle is a read-write cycle and the data out will contain data read from the selected cell. If neither of the above sets of conditions is satisfied, the condition of the data out (at access time) is indeterminate.

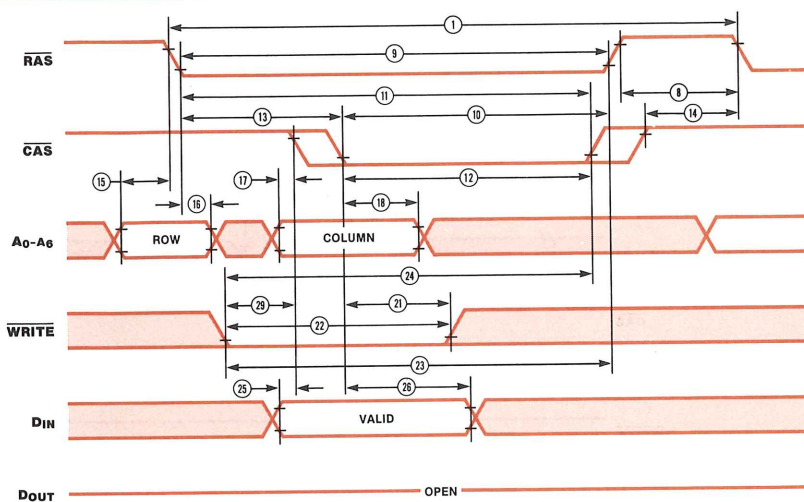
17. Effective capacitance calculated from the equation $C = I\Delta V/\Delta V = 3$ V and power supplies at nominal levels.

18. $\overline{CAS} = V_{IH}$ to disable D_{OUT} .

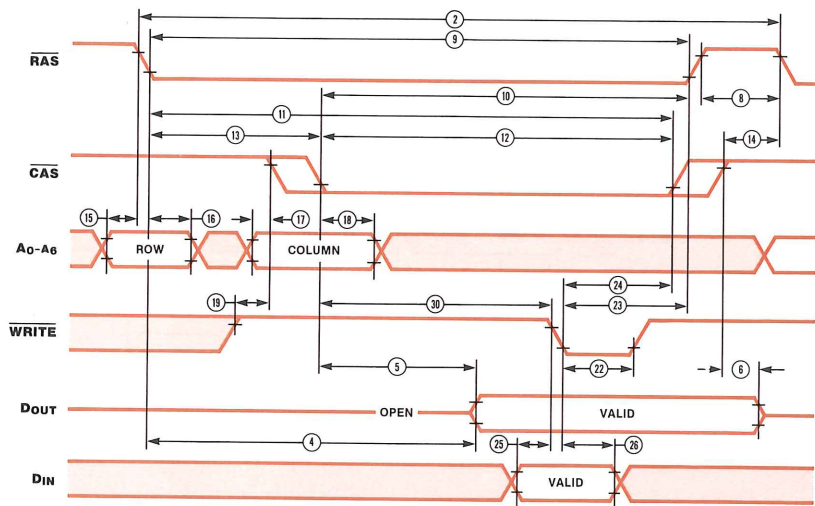
Read Cycle



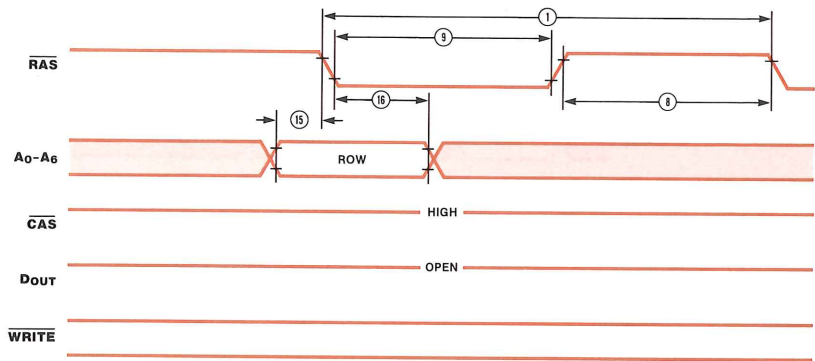
Write Cycle



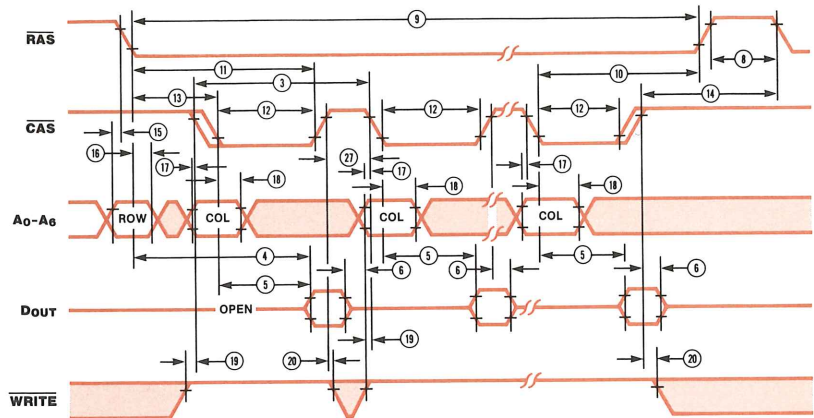
Read/Modify/ Write Cycle



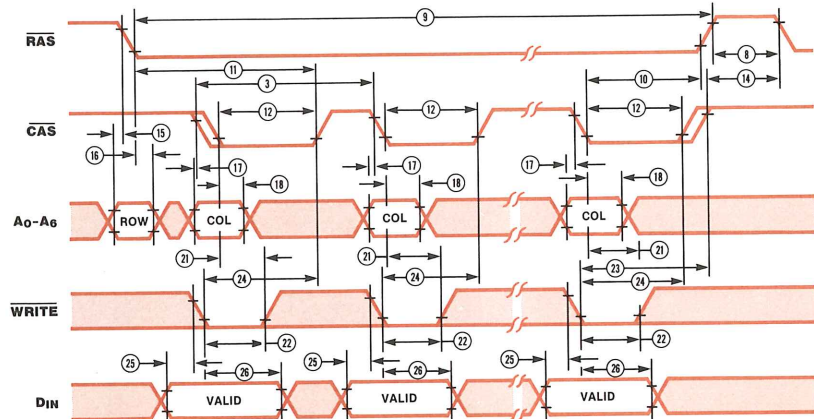
RAS-Only Refresh Cycle



Page-Mode
Read
Cycle



Page-Mode
Write
Cycle



Z6116 16K x 1 Dynamic RAM



NEW
1980

Product Specification

February 1980

Features

- 16,384 words by 1-bit organization
- Industry-standard 16-pin package
- $\pm 10\%$ tolerance on all power supplies
- Low power consumption: 462 mW active; 20mW standby (max)
- Output data controlled by $\overline{\text{CAS}}$ and unlatched at end of cycle
- Common I/O capability using the early-write mode

- Read/modify/write, page-mode and RAS-only refresh capabilities
- On-chip address and data-in latches
- 2 ms refresh period, 128-cycle refresh

	Z6116-2	Z6116-3	Z6116-4
Maximum Access Time (ns)	150	200	250
Read, Write Cycle (ns)	320	375	410
Read/Modify/Write Cycle (ns)	320	405	500

General Description

The Z6116 is a 16,384-bit, high-speed dynamic random-access memory designed for high-performance, low-cost applications in mainframe and buffer memories and peripheral storage. Organized as 16,384 one-bit words and fabricated using a double-poly n-channel silicon-gate process, this device optimizes speed, power and density tradeoffs.

By multiplexing row and column address inputs, the Z6116 requires only seven address lines and permits packaging in a standard 16-pin dual in-line package. This packaging technique allows high system density and is compatible with widely available automated test and insertion equipment. Complete address decoding is done on chip.

All inputs are TTL compatible, and the output is 3-state TTL compatible. On-chip address

and data registers eliminate the need for interface registers.

The single transistor storage cell requires refreshing for data retention. Refreshing is accomplished by performing $\overline{\text{RAS}}$ -only refresh cycles, hidden refresh cycles, or normal read or write cycles on the 128 address combinations of A_0 through A_6 during a 2 ms period. A write cycle refreshes stored data on all bits of the selected row except the bit which is addressed.

In addition to the usual read, write and read/modify/write cycles, the Z6116 is capable of delayed write cycles, page-mode operation and $\overline{\text{RAS}}$ -only refresh. Furthermore, proper control of $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and $\overline{\text{WRITE}}$ permits common I/O capability, two-dimensional chip selection and extended page boundaries.

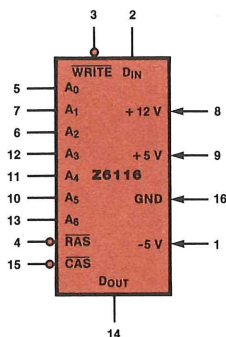


Figure 1. Logic Symbol

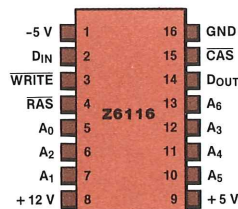


Figure 2. Pin Assignments

A_0-A_6	Address Inputs
D_IN	Data Input
D_OUT	Data Output
CAS	Column Address Strobe
RAS	Row Address Strobe
WRITE	Read/Write Input
-5 V	-5 V Power Supply (V_{BB})
+5 V	+5 V Power Supply (V_{CC})
+12 V	+12 V Power Supply (V_{DD})
GND	Ground (V_{SS})

Pin Names

**Functional
Description**

Addressing. The fourteen address bits required to decode one of the 16,384 cell locations within the Z6116 are multiplexed onto the seven address inputs and latched into the on-chip address latches by externally applying two negative-going TTL-level clocks. The first clock, Row Address Strobe ($\overline{\text{RAS}}$), latches the seven row address bits into the chip. The second clock, Column Address Strobe ($\overline{\text{CAS}}$), triggers a sequence of events that are controlled by different delayed internal clocks. The two clock chains are linked together logically so that the address multiplexing operation is done outside the critical-path timing sequence for read data access. The later events in the $\overline{\text{CAS}}$ clock sequence are inhibited until the occurrence of a delayed signal derived from the $\overline{\text{RAS}}$ clock chain.

This "gated $\overline{\text{CAS}}$ " feature allows the $\overline{\text{CAS}}$ clock to be externally activated as soon as the Row Address Hold Time specification has been satisfied and the seven address inputs have been changed from row address to column address information.

$\overline{\text{CAS}}$ can be activated any time after t_{RAH} with no effect on the worst-case data access time (t_{RAC}), up to the point when the delayed row clock no longer inhibits the remaining sequence of column clocks.

Two timing end points result from the internal gating of $\overline{\text{CAS}}$ and are called $t_{\text{RCD}(\text{min})}$ and $t_{\text{RCD}(\text{max})}$. No data or storage errors result if $\overline{\text{CAS}}$ is applied to the Z6116 any time beyond the $t_{\text{RCD}(\text{max})}$ limit. However, access time is then determined exclusively by the access time from $\overline{\text{CAS}}$ (t_{RAC}), and the access time from $\overline{\text{RAS}}$ is lengthened by the amount that t_{RCD} exceeds the $t_{\text{RCD}(\text{max})}$ limit.

Data Input/Output. An on-chip register latches information on the Data In pin whenever $\overline{\text{RAS}}$ is active and both $\overline{\text{CAS}}$ and $\overline{\text{WRITE}}$ have made their negative transitions. The later of the two signals ($\overline{\text{CAS}}$ or $\overline{\text{WRITE}}$) that makes its negative transition generates the strobe for the Data In register.

This feature is illustrated by referencing D_{IN} to $\overline{\text{WRITE}}$ in the timing diagrams that depict the read/write and page-mode write cycles, while the early-write cycle diagram shows D_{IN} referenced to $\overline{\text{CAS}}$.

In the read cycle, data is retrieved from the memory by maintaining $\overline{\text{WRITE}}$ inactive throughout the portion of the memory cycle in which $\overline{\text{CAS}}$ is active. Data then appears at the output within the specified access time.

Unlike most 16-pin 4K dynamic RAMs and some 16K dynamic RAMs, the Data Output pin of the Z6116 is normally in the high-impedance state ($\overline{\text{CAS}}$ inactive). The only time the output buffer turns on and presents a logic 0 or 1 at the D_{OUT} pin is at access time during a read, read/modify/write or delayed-write cycle. The output remains valid from access time until $\overline{\text{CAS}}$ returns High. The output data is not inverted (that is, it has the same polarity as the input data).

In the early-write cycle, however, the output pin stays in the high-impedance state throughout the entire cycle.

This type of $\overline{\text{CAS}}$ -controlled latch provides features that are important in the following system applications:

- **Common I/O Operation.** If all write operations are performed in the early-write mode, then D_{IN} and D_{OUT} can be tied together for a common I/O data bus.
- **Flexible $\overline{\text{RAS}}/\overline{\text{CAS}}$ Clock Timing.** D_{OUT} is controlled only by $\overline{\text{CAS}}$. It remains valid during a read cycle from access time until $\overline{\text{CAS}}$ returns High. The data can be maintained valid from one cycle until a new memory cycle begins with no penalty in cycle time. Thus, $\overline{\text{CAS}}$ and $\overline{\text{RAS}}$ can return to their inactive states independent of each other, provided the t_{RAS} and t_{CAS} specifications are met.
- **Two Methods of Chip Selection.** Since $\overline{\text{CAS}}$ is not required to turn off the outputs of unselected memory devices in a matrix (there is no special output deselect cycle), both $\overline{\text{CAS}}$ and/or $\overline{\text{RAS}}$ can be decoded for chip selection. If both $\overline{\text{CAS}}$ and $\overline{\text{RAS}}$ are decoded, a two-dimensional (X, Y) chip-selected array can be constructed.
- **Extended Page Boundary.** If $\overline{\text{CAS}}$ is decoded as a page-select signal, the page boundary can be extended beyond the 128 column locations in a single chip.

Functional Description
(Continued)

Page Mode Operation. Page mode operation allows successive memory cycles at multiple column locations of the same row address. This mode is achieved by latching the row address into the chip and maintaining $\overline{RAS}$ active throughout all successive memory cycles in which the row address is unchanged. This mode of operation results in power savings (power associated with the negative-going edge of $\overline{RAS}$ is not dissipated), and improved access and cycle times (time required for strobing in a row address is eliminated).

The page boundary of a single Z6116 is limited to the 128 column locations determined by all combinations of the seven address bits. However, in system applications that require more than 16K words, the page boundary can be extended by using $\overline{\text{CAS}}$ rather than $\overline{\text{RAS}}$ as

the chip-select signal. $\overline{\text{RAS}}$ is applied to all devices to latch the row address into each device. $\overline{\text{CAS}}$ is then decoded and serves as a page-cycle select signal. Only those devices that receive both $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ execute a read or write cycle.

Refresh. The dynamic cell matrix is refreshed with a memory cycle at each of the 128 row addresses within each 2 ms time interval. Although any normal memory cycle accomplishes refresh, this function is best done with $\overline{\text{RAS}}$ -only cycles, because $\overline{\text{RAS}}$ -only refresh results in a substantial reduction in operating power. During standby operation, V_{CC} power may be removed without affecting refresh operation.



Figure 3. Block Diagram

Power Considerations

Because most of the circuitry on the Z6116 is dynamic, power dissipation is primarily a function of operating frequency rather than duty cycle. In applications requiring lower power dissipation, the device can be operated at a lower frequency (longer cycle time) to reduce the average power dissipation.

Although $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ can be decoded and used as a chip select signal for the Z6116, overall system power is minimized if only $\overline{\text{RAS}}$ is used for this purpose. All unselected devices (those that do not receive $\overline{\text{RAS}}$) remain in a low-power (standby) mode, regardless of the state of $\overline{\text{CAS}}$.

Although no particular power supply noise restrictions exist, other than that the supply voltages must remain within the specified tolerance limits, adequate decoupling should be provided to suppress high-frequency noise

resulting from the transient current of the device. This insures optimum system performance and reliability. Bulk capacitance requirements are minimal since the Z6116 draws very little dc current.

The Z6116 requires no special power supply sequencing as long as the Absolute Maximum Rating conditions are observed. To insure compliance with the Absolute Maximum Ratings, Zilog recommends that -5 V be applied first and removed last. The -5 V connection must never be more positive than GND when power is applied to the $+12\text{ V}$ connection.

After power is applied to the device, the Z6116 requires several cycles before proper operation of the device is achieved. Any eight cycles that perform refresh satisfy this requirement.

Absolute Maximum Ratings

Voltage on any pin relative to V_{BB} -0.5 V to $+20\text{ V}$
 Voltage on V_{DD} , V_{CC} supplies relative to V_{SS} -1.0 V to $+15.0\text{ V}$
 $V_{\text{BB}} - V_{\text{SS}}$ ($V_{\text{DD}} - V_{\text{SS}} > 0\text{ V}$) 0 V
 Ambient operating temperature, T_{A} 0°C to $+70^{\circ}\text{C}$
 Ambient storage temperature -65°C to $+150^{\circ}\text{C}$

Short circuit output current 50 mA
 Power dissipation 1 Watt

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

All ac parameters assume a load capacitance of 100 pF max. Timing references between two output signals assume a load difference of 50 pF max. Standard conditions are as follows:

- $+4.5\text{ V} \leq V_{\text{CC}} \leq +5.5\text{ V}$
- $+10.8\text{ V} \leq V_{\text{DD}} \leq +13.2\text{ V}$
- $-5.5\text{ V} \leq V_{\text{BB}} \leq -4.5\text{ V}$
- $\text{GND} = 0\text{ V}$
- $0^{\circ}\text{C} \leq T_{\text{A}} \leq +70^{\circ}\text{C}$

Capacitance	Symbol	Parameter	Typ	Max	Units	Notes
	C_{11}	Input Capacitance ($A_0 - A_6$), D_{IN}	4	5	pF	17
	C_{12}	Input Capacitance $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, $\overline{\text{WRITE}}$	8	10	pF	17
	C_0	Output Capacitance (D_{OUT})	5	7	pF	17, 18

DC Character-istics	Symbol	Parameter	Min	Max	Unit	Notes
	V _{DD}	Supply Voltage	10.8	13.2	V	2
	V _{CC}	Supply Voltage	4.5	5.5	V	2, 3
	V _{SS}	Supply Voltage	0	0	V	2
	V _{BB}	Supply Voltage	-4.5	-5.5	V	2
	V _{IH}	Input High (Logic 1) Voltage all inputs	2.4	7.0	V	2
	V _{IL}	Input Low (Logic 0) Voltage all inputs	-1.0	0.8	V	2
	I _{DD1}	OPERATING CURRENT		35	mA	4
	I _{CC1}	Average power supply operating current (RAS, CAS cycling; t _{RC} = min. spec.)		200	μA	5
	I _{DD2}	STANDBY CURRENT		1.5	mA	
	I _{CC2}	Power supply standby current	-10	10	μA	
	I _{BB2}	(RAS = V _{IH} , D _{OUT} = High impedance)		100	μA	
	I _{DD3}	REFRESH CURRENT		27	mA	4
	I _{CC3}	Average power supply current, refresh mode	-10	10	μA	
	I _{BB3}	(RAS cycling, CAS = V _{IH} t _{RC} = min. spec.)		200	μA	
	I _{DD4}	PAGE MODE CURRENT		27	mA	4
	I _{CC4}	Average power supply current, page-mode operation (RAS = V _{IH} , CAS cycling; t _{PC} = min. spec.)		200	μA	5
	I _{IL}	INPUT LEAKAGE Input leakage current, any input (V _{BB} = -5 V, 0 V ≤ V _{IN} ≤ +7.0 V, all other pins not under test = 0 V)	-10	10	μA	
	V _{OL}	OUTPUT LEAKAGE Output leakage current (D _{OUT} is disabled, 0 V ≤ V _{OUT} ≤ +5.5 V)	-10	10	μA	
	V _{OH}	OUTPUT LEVELS Output high (Logic 1) voltage (I _{OUT} = -5 mA)	2.4		V	3
	V _{OL}	Output low (Logic 0) voltage (I _{OUT} = 4.2 ma)		0.4	V	

NOTES:

1. T_A is specified for operation at frequencies to t_{RC} ≥ t_{RC} (min). Operation at higher cycle rates with reduced ambient temperatures and higher power dissipation is permissible, provided ac operating parameters are met.
2. All voltages referenced to GND.
3. Output voltage will swing from GND to V_{CC} when activated with no current loading. For purposes of maintaining data in standby mode, V_{CC} may be reduced to V_{SS} without affecting refresh operations or data retention. However, the V_{OH} (min) specification is not guaranteed in this mode.
4. I_{DD1}, I_{DD3} and I_{DD4} depend on cycle rate.
5. I_{CC1} and I_{CC4} depend upon output loading. During readout of high level data V_{CC} is connected through a low impedance (125Ω typ) to data out. At all other times I_{CC} consists of leakage currents only.

AC

Character- istics 6, 7, 8

Number	Symbol	Parameter	Z6116-2*		Z6116-3*		Z6116-4*		Notes
			Min	Max	Min	Max	Min	Max	
1	t_{RC}	Random Read or Write Cycle Time	320		375		410		9
2	t_{RMW}	Read/Modify/Write Cycle Time	320		405		500		9
3	t_{PC}	Page Mode Cycle Time	170		225		275		
4	t_{RAC}	Access Time from $\overline{RAS}$		150		200		250	10, 12
5	t_{CAC}	Access Time from $\overline{CAS}$		100		135		165	11, 12
6	t_{OFF}	Output Buffer Turn-Off Delay	0	40	0	50	0	60	13
7	t_T	Transition Time (rise and fall)	3	35	3	50	3	50	8
8	t_{RP}	$\overline{RAS}$ Precharge Time	100		120		150		
9	t_{RAS}	$\overline{RAS}$ Pulse Width	150	10K	200	10K	250	10K	
10	t_{RSH}	$\overline{RAS}$ Hold Time	100		135		165		
11	t_{CSH}	$\overline{CAS}$ Hold Time	150		200		250		
12	t_{CAS}	$\overline{CAS}$ Pulse Width	100	10K	135		165		
13	t_{RCD}	$\overline{RAS}$ to $\overline{CAS}$ Delay Time	20	50	25	65	35	85	14
14	t_{CRP}	$\overline{CAS}$ to $\overline{RAS}$ Precharge Time	-20		-20		-20		
15	t_{ASR}	Row Address Set-Up	0		0		0		
16	t_{RAH}	Row Address Hold Time	20		25		35		
17	t_{ASC}	Column Address Set-Up Time	-10		-10		-10		
18	t_{CAH}	Column Address Hold Time	45		55		75		
19	t_{RCS}	Read Command Set-Up Time	0		0		0		
20	t_{RCH}	Read Command Hold Time	0		0		0		
21	t_{WCH}	Write Command Hold Time	45		55		75		
22	t_{WP}	Write Command Pulse Width	45		55		75		
23	t_{RWL}	Write Command to $\overline{RAS}$ Lead Time	50		70		85		
24	t_{CWL}	Write Command to $\overline{CAS}$ Lead Time	50		70		85		
25	t_{DS}	Data In Set-Up Time	0		0		0		15
26	t_{DH}	Data In Hold Time	45		55		75		15
27	t_{CP}	$\overline{CAS}$ Precharge (page mode only)	60		80		100		
28	t_{REF}	Refresh Period		2ms		2ms		2ms	
29	t_{WCS}	$\overline{WRITE}$ Command Set-Up Time	-20		-20		-20		16
30	t_{CWD}	$\overline{CAS}$ to $\overline{WRITE}$ Delay	60		80		90		16
31	t_{RWD}	$\overline{RAS}$ to $\overline{WRITE}$ Delay	110		145		175		16

6. Several cycles are required after power-up before proper device operation is achieved. Any 8 cycles which perform refresh are adequate for this purpose.

7. AC measurements assume $t_T = 5$ ns.

8. V_{IH} (min) and V_{IL} (max) are reference levels for measuring timing of input signals. Also, transition times are measured between V_{IH} and V_{IL} .

9. The specifications for t_{RC} (min) and t_{RWC} (min) are used only to indicate cycle time at which proper operation over the full temperature range ($0^\circ\text{C} \leq T_A \leq 70^\circ\text{C}$) is assured.

10. Assumes that $t_{RCD} \leq t_{RCD}(\text{max})$. If t_{RCD} is greater than the maximum recommended value shown in this table, t_{RAC} increases by the amount that t_{RCD} exceeds the value shown.

11. Assumes that $t_{RCD} \geq t_{RCD}(\text{max})$.

12. Measured with a load equivalent to 2 TTL loads and 100 pF.

13. $t_{OFF}(\text{max})$ defines the time at which the output achieves the open circuit condition and is not referenced to output voltage levels.

14. Operation within the $t_{RCD}(\text{max})$ limit insures that $t_{RAC}(\text{max})$ can be met. $t_{RCD}(\text{max})$ is specified as a reference point only; if t_{RCD} is greater than the specified $t_{RCD}(\text{max})$ limit, then access time is controlled exclusively by t_{CAC} .

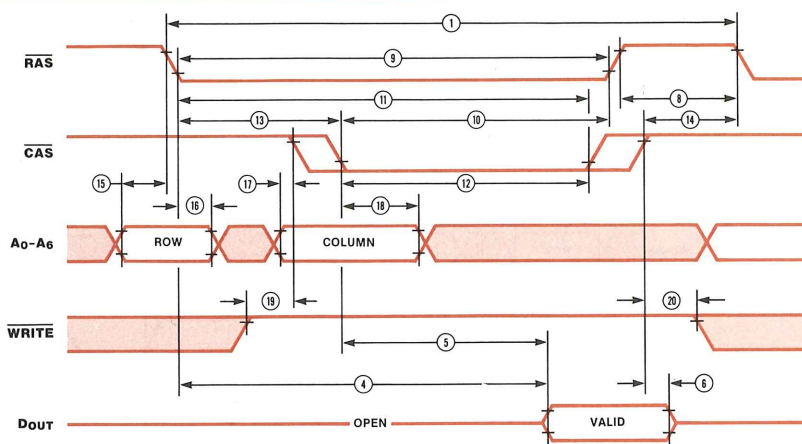
15. These parameters are referenced to $\overline{CAS}$ leading edge in early-write cycles and to $\overline{WRITE}$ leading edge in delayed write or read/modify/write cycles.

16. t_{WCS} , t_{CWD} , and t_{RWD} are not restrictive operating parameters. They are included in the data sheet as electrical characteristics only: if $t_{WCS} \geq t_{WCS}(\text{min})$, the cycle is an early-write cycle and the data out pin will remain open circuit (high impedance) throughout the entire cycle. If $t_{CWD} \geq t_{CWD}(\text{min})$ and $t_{RWD} \geq t_{RWD}(\text{min})$, the cycle is a read-write cycle and the data out will contain data read from the selected cell. If neither of the above sets of conditions is satisfied, the condition of the data out (at access time) is indeterminate.

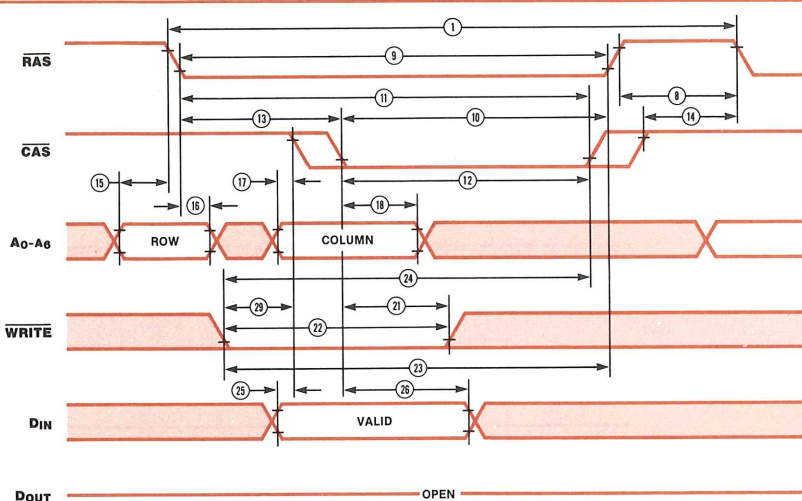
17. Effective capacitance calculated from the equation $C = I\Delta t/\Delta V = 3$ V and power supplies at nominal levels.

18. $\overline{CAS} = V_{IH}$ to disable D_{OUT} .

Read Cycle

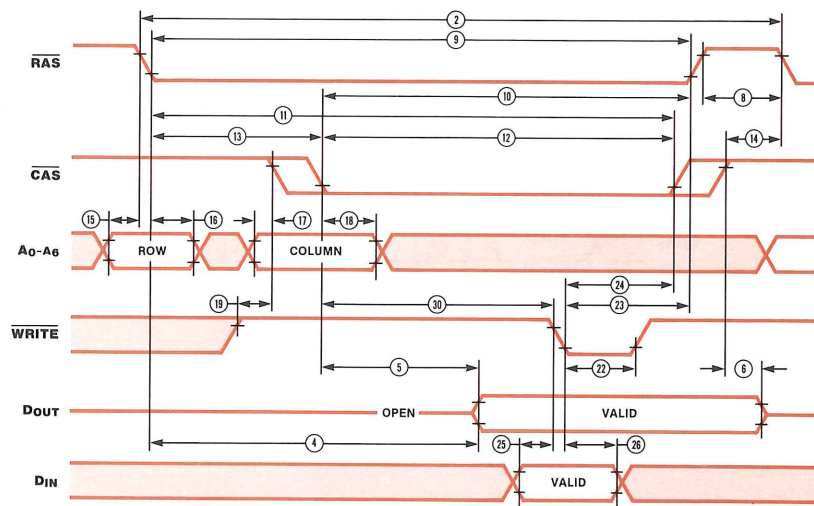


Write Cycle

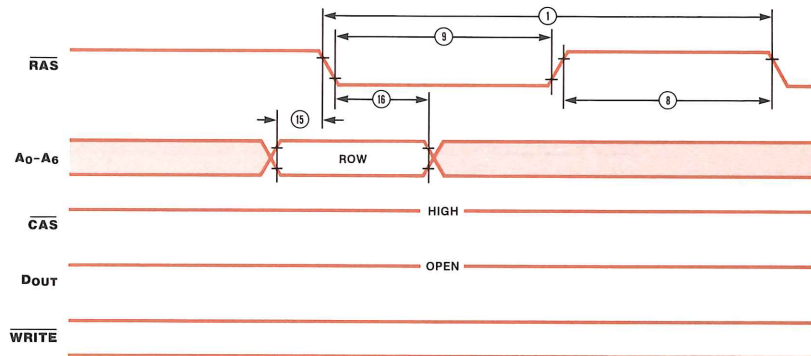


3

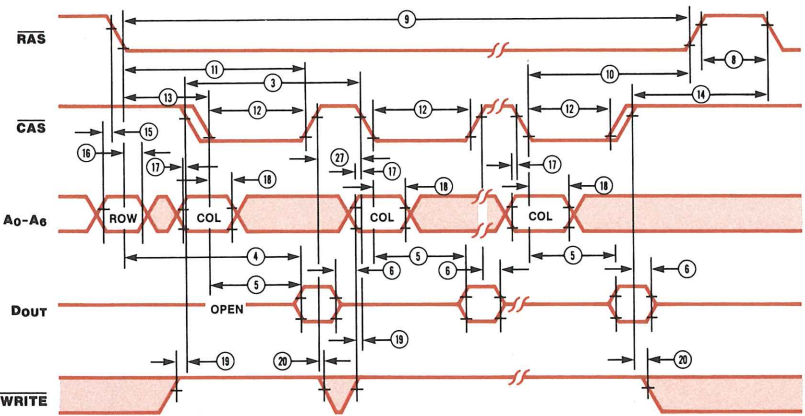
Read/Modify/ Write Cycle



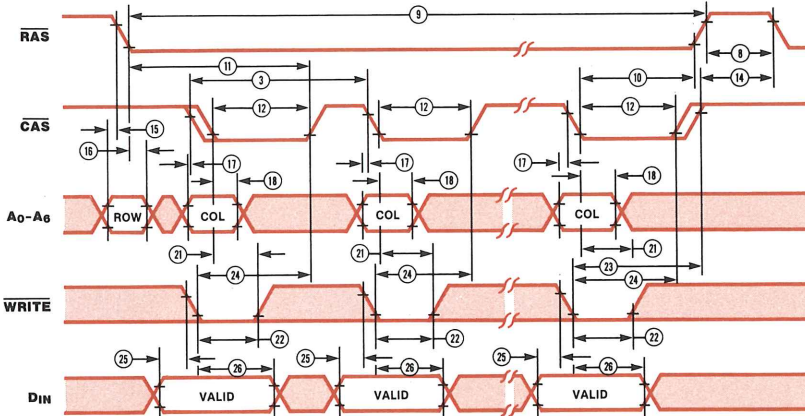
RAS-Only Refresh Cycle



Page-Mode
Read
Cycle



Page-Mode
Write
Cycle



Z6132 4K x 8 Quasi-Static RAM



**NEW
1980**

Product Specification

Preliminary

February 1980

Description

The Zilog Z6132 is a +5 V intelligent MOS dynamic RAM organized as 4096 words by eight bits. Although it uses single-transistor dynamic storage cells, the Z6132 effectively functions as a static RAM because it performs and controls its own refresh. This eliminates the need for external refresh circuitry and combines the convenience of a static RAM with the high density and low power consumption normally associated with a dynamic RAM.

The Z6132 is particularly suited for microprocessor and minicomputer applications where its byte-wide organization, transparent self-refresh and single supply voltage reduce the parts count and simplify the design.

The Z6132 uses high-performance depletion-load double-poly n-channel silicon-gate MOS technology with a mixture of static and dynamic circuitry that provides a small memory cell, fast access and low power consumption. The Z6132 has separate pins for addresses and bidirectional data I/O to provide maximum flexibility in its application.

The circuit is packaged in an industry-standard 28-pin DIP and pin compatible with the proposed JEDEC standard.

The Z6132 conforms with the Z-Bus specification used by the new generation of Zilog microprocessors, the Z8 and Z8000.

Features

- Byte-wide organization: 4096 words by eight bits
 - Access and cycle times guaranteed over voltage and temperature range:
- | Part Number | Access Time | Cycle Time |
|-------------|-------------|------------|
| Z6132-3 | 200 ns | 350 ns |
| Z6132-4 | 250 ns | 375 ns |
| Z6132-5 | 300 ns | 425 ns |
- Low power consumption: 200 mW active, 125 mW stand-by

- Industry-standard 28-pin DIP with JEDEC-recommended pinout
- $\pm 10\%$ tolerance on single +5 V supply voltage
- Automatic self-refresh scheme with slow-and-fast-cycle modes
- All inputs and outputs are TTL compatible
- On-chip substrate bias generator
- Interfaces readily to Z8 and Z8000

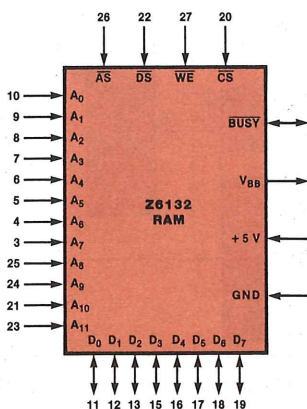


Figure 1. Logic Symbol

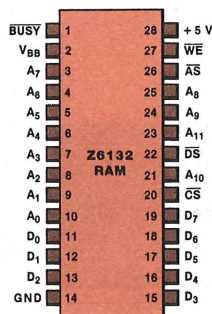


Figure 2. Pin Assignments

A₀-A₁₁	Address inputs
D₀-D₇	Data Inputs/Outputs (3-state)
AC	Address Clock input (rising edge)
DS	Data Strobe input (active Low)
WE	Write Enable input (active Low)
CS	Chip Select input (active Low)
BUSY	Busy output (active Low; open drain) and Refresh Mode Control input
VBB	Negative Substrate Bias output
VCC	+5 V supply connection
VSS	0 V = Ground connection

Pin Names

Functional Description

The Z6132 4K x 8 quasi-static RAM is organized as two separate blocks, each having two sets of 64 rows on either side of the 128 sense amplifiers (Figure 3). Both blocks have separate and independent row address buffers and decoders, but they share the column decoder and the internal 8-bit wide data path. The two sets of row address decoders are addressed either by the address inputs A_1 – A_7 or by the internal 7-bit refresh counter. The least significant address input (A_0) selects one of the two blocks for external access. While the selected block performs a read or write operation, the other memory block uses the refresh counter address to refresh one row. Details of the self-refresh mechanism are explained later.

A memory cycle starts when the rising edge

of Address Clock (AC) clocks in Chip Select ($\overline{CS}$), A_0 , and Write Enable ($\overline{WE}$). If the chip is not selected ($\overline{CS} = \text{High}$), all other inputs are ignored for the rest of the cycle (that is, until the next rising edge of AC). Both memory blocks are self refreshed by the 7-bit refresh counter. If the chip is selected ($\overline{CS} = \text{Low}$), the 12 address bits and the Write Enable bit are clocked into their registers. A_0 determines which block is addressed by A_1 – A_{11} ; the other block is refreshed by the 7-bit refresh counter.

The Chip Select and Address inputs must be valid only during a short hold time after the rising edge of AC. This allows address/data multiplexing, because data I/O is controlled by a separate control input Data Strobe ($\overline{DS}$).

Read Cycle

A read cycle is initiated by the rising edge of Address Clock (AC) while Chip Select ($\overline{CS}$) is Low and Write Enable ($\overline{WE}$) High. A Low level on the Data Strobe ($\overline{DS}$) input activates the Data outputs after a specified delay from the

rising edge of AC as well as the falling edge of $\overline{DS}$, whichever comes later. During a read operation, $\overline{DS}$ is nothing but a static Output Enable signal.

Write Cycle

A write cycle is initiated by the rising edge of Address Clock (AC) while Chip Select ($\overline{CS}$) is Low and Write Enable ($\overline{WE}$) is Low.

The $\overline{WE}$ input is checked again at the beginning (falling edge) of Data Strobe ($\overline{DS}$).

If $\overline{WE}$ is still Low, this falling edge of $\overline{DS}$ edge-triggers the data on the D_0 – D_7 inputs into the addressed memory location. Data must be valid only during a short hold time after the falling edge of $\overline{DS}$.

Write Inhibit Cycle

After a write cycle has been initiated, the actual write operation can still be aborted by pulling $\overline{WE}$ High again before the falling edge of $\overline{DS}$. This write inhibit cycle is a special feature that permits starting a write cycle early

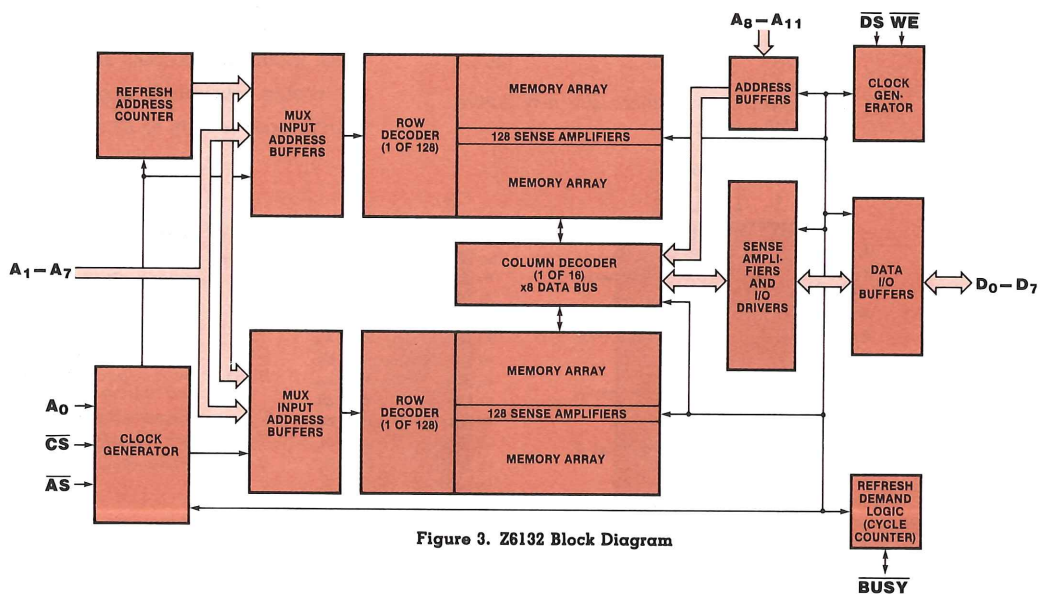
at AC time, but still allows the option of inhibiting the write operation later at $\overline{DS}$ time.

Note: Whenever a write cycle has been initiated, it must be accompanied by a High-to-Low transition on the Data Strobe input.

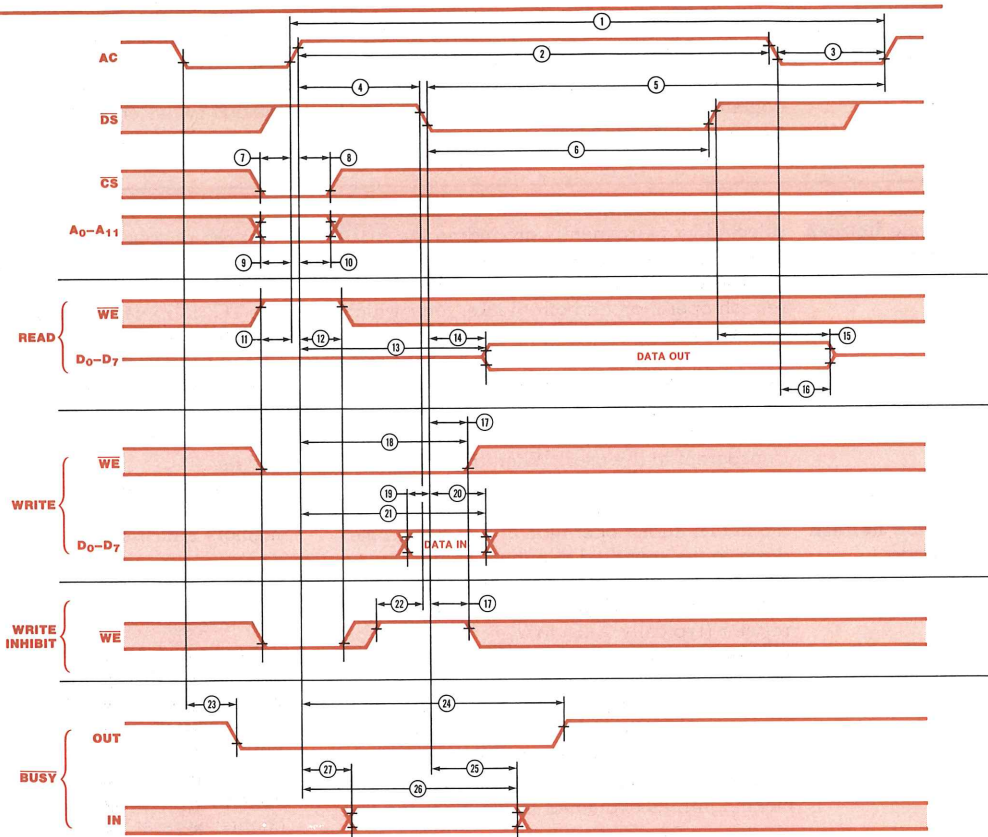
Maximum Cycle Time

The maximum read or write cycle time requirements (15,000 and 800 ns) do not apply to

any individual cycle. They are specified to guarantee a complete refresh in a 2 ms period.



AC
Electrical
Character-
istics



No.	Symbol	Parameter	Z6132-3		Z6132-4		Z6132-5		Unit	Comments
			Min	Max	Min	Max	Min	Max		
1	TC	Read or Write Cycle Time	600	15000	650	15000	700	15000	ns	BUSY = Low
2	TwACh	AC Width (High)	350	800	375	800	425	800	ns	BUSY = High
3	TwACl	AC Width (Low)	480		525		580		ns	BUSY = Low
4	TdAC(DS)	AC ↑ to DS ↓	230		260		280		ns	BUSY = High
5	TdDS(AC)	DS ↓ to AC ↑	40		50		60		ns	
6	TwDS	DS Width (Low)	510		540		570		ns	BUSY = Low
7	TsCS(AC)	CS Setup Time to AC ↑	240		250		260		ns	BUSY = High
8	ThCS(AS)	CS Hold Time to AC ↑	120		140		160		ns	
9	TsA(AC)	Address Setup Time to AC ↑	0		0		0		ns	
10	ThA(AC)	Address Hold Time to AC ↑	40		50		60		ns	
11	TsW(AC)	WE Setup Time to AC ↑	0		0		0		ns	
12	ThW(AC)	WE Hold Time to AC ↑	0		0		0		ns	
13	TdAC(DO)	AC ↑ to Data Out	60		70		80		ns	
14	TdDS(DO)	DS ↓ to Data Out	200		250		300		ns	Which ever is later
15	TdDS(DOz)	DS ↓ to Data Out Float	70		80		90		ns	
16	TdAC(DOz)	AC ↓ to Data Out Float	30		35		40		ns	Which ever is earlier
17	ThW(DS)	WE Hold Time to DS ↓	30		35		40		ns	Which ever is later
18	ThW(AC)	WE Hold Time to AC ↑	60		70		80		ns	
19	TsDI(DS)	Data In Setup Time to DS ↓	120		130		140		ns	
20	ThDI(DS)	Data In Hold Time to DS ↓	0		0		0		ns	
21	ThDI(AC)	Data In Hold Time to AC ↑	50		60		70		ns	Which ever is later
22	TsWh(DS)	WE High Setup Time to DS ↓	120		130		140		ns	
23	TdAC(BI)	AC ↓ to BUSY Out ↓	10		10		10		ns	
24	TdAC(Bh)	AC ↑ to BUSY Out ↑	80		90		100		ns	Selected
25	ThB(DS)	BUSY In Hold Time to DS ↓	400		450		500		ns	Deselected
26	ThB(AC)	BUSY In Hold Time to AC ↑	80		90		100		ns	Which ever is later
27	TsB(AC)	BUSY In Setup Time to AC ↑	70		80		90		ns	

Self-Refresh Operation

The Z6132 stores data in single-transistor dynamic cells that must be refreshed at least every 2 ms. Each of the two memory blocks contains 16,384 cells and requires 128 refresh cycles to completely refresh the array.

The Z6132 operates in one of two user-

selectable self-refresh modes, each satisfying the refresh time requirements. On the basis of the available memory cycle time, the user can decide to use either the Long Cycle-Time Refresh Mode or the Short Cycle-Time Refresh Mode.

Long Cycle-Time Refresh Mode

This is the simplest self-refresh mode, and is selected by permanently grounding the $\overline{\text{BUSY}}$ output pin. Every memory cycle in this mode consists of a memory operation followed by a refresh operation on both blocks, after which the refresh counter is incremented. Internally, the complete cycle consists of a 4-phase sequence: 1. Memory read, write, or write inhibit. 2. Precharge. 3. Refresh. 4. Precharge. These internal operations are automatic and transparent to the user. When the chip is not

selected ($\overline{\text{CS}} = \text{High}$ when AC goes High), the first two phases are omitted.

There are two important requirements: the memory cycle time must always be longer than the TC (Min) value specified for $\overline{\text{BUSY}} = \text{Low}$ and there must be at least 128 Address Clocks in any 2 ms period.

The Long Cycle-Time Refresh mode is the one most practical for microprocessor applications, where the cycle time usually exceeds 700 ns.

Short Cycle-Time Refresh Mode

This is a more sophisticated self-refresh mode that allows operation at any cycle time down to the specified minimum value.

The user selects this mode by pulling the $\overline{\text{BUSY}}$ output pin High through a pull-up resistor (typically 1 k Ω) to V_{CC} . The $\overline{\text{BUSY}}$ outputs of several Z6132 chips can be or-tied together.

In this mode, the Z6132 always performs a refresh operation on the memory block that is not being addressed from the outside. The refresh counter is incremented whenever it is meaningful, as explained in the following text.

Deselect Self-Refresh. If the chip is deselected ($\overline{\text{CS}} = \text{High}$ when AC goes High), both blocks are refreshed and the refresh counter is incremented after every cycle.

Odd/Even Self-Refresh. If the chip is selected ($\overline{\text{CS}} = \text{Low}$ when AC goes High), the refresh counter refreshes the block that is not addressed by A_0 . The refresh counter is incremented after an even and an odd address have occurred. This self-refresh scheme takes advan-

tage of the inherent sequential nature of most memory addressing.

Cycle-Count Self-Refresh. Normally the deselect and odd/even self-refresh schemes step through 128 refresh addresses in less than 2 ms. To guarantee proper refresh operation even in the exceptional case when the memory is continually selected and addressed by a long string of all even or all odd addresses, a built-in cycle counter activates the $\overline{\text{BUSY}}$ output and requests one longer memory cycle to append a refresh operation. This internal cycle counter is reset whenever the refresh counter is incremented. The cycle counter then counts memory cycles and activates the $\overline{\text{BUSY}}$ output when it reaches a count of 18.

$\overline{\text{BUSY}}$ is fed into the $\overline{\text{WAIT}}$ input of most microprocessors. $\overline{\text{BUSY}}$ is a request to the CPU for a longer memory cycle and is kept Low until the refresh cycle has started. $\overline{\text{BUSY}}$ only becomes active when the Z6132 has been selected and addressed with all odd or all even addresses for 18 consecutive Address Strokes.

Mixed Cycle Time Refresh Mode

External logic can be used to select between Long and Short Cycle Time Refresh modes by controlling the $\overline{\text{BUSY}}$ pin as an input. The Timing Diagram (parameters 25 through 27) shows when the internal logic interrogates the $\overline{\text{BUSY}}$ input.

When $\overline{\text{BUSY}}$ is Low the cycle must be long, both blocks are refreshed and the refresh counter is incremented every cycle.

When $\overline{\text{BUSY}}$ is High, the cycle can be short and the refresh operation is performed as described under Short Cycle Time Refresh Mode.

The external logic must guarantee proper refresh timing. If the Z6132 received a sequence of 17 consecutive all odd or all even addresses while it was continuously selected and $\overline{\text{BUSY}}$ was held High, the $\overline{\text{BUSY}}$ output will go Low as described before.

A current limiting resistor of $\sim 1\text{k}\Omega$ should

be inserted if the $\overline{\text{BUSY}}$ pin is driven by TTL logic.

External logic, as shown in Figure 4 can detect the fact that the memory requires a long cycle time and can pull the CPU $\overline{\text{WAIT}}$ input Low.

Note that the cycle time in most microprocessor applications is so long that the simple Long Cycle Time Refresh Mode is sufficient.

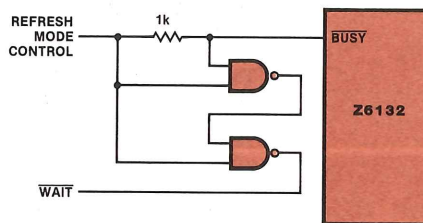
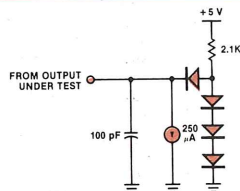


Figure 4. External $\overline{\text{WAIT}}$ Generation

Substrate Bias Generator	The Z6132 contains an on-chip negative substrate-bias generator, which is a simple dc-to-dc converter that generates a substrate-bias voltage of -2.5 to -3 V. This reduces parasitic	junction capacitances and thus increases circuit speed. The substrate bias output V_{BB} should be decoupled externally with an $\approx 0.1 \mu\text{F}$ ceramic capacitor to V_{SS} (ground).
Power-Up	After applying V_{CC} , it is necessary to wait 20 ms to charge the substrate bias decoupling capacitor. Moreover, the 6132 requires sixteen	selected or deselected memory cycles before proper operation is attained.
Absolute Maximum Ratings	Voltages on all pins (except V_{BB}) with respect to GND -0.5 V to +7.0 V Operating Ambient Temperature 0°C to +70°C Storage Temperature -65°C to +150°C	Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.
Standard Test Conditions	The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:	□ $+4.5 \text{ V} \leq V_{CC} \leq +5.5 \text{ V}$ □ $V_{SS} = \text{GND} = 0 \text{ V}$ □ $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$

DC Electrical Characteristics	Symbol	Parameter	Min	Max	Unit	Condition
	V_{IH}	Input High Voltage	2.0	7.0	V	
	V_{IL}	Input Low Voltage	-0.5	0.8	V	
	V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -3 \text{ mA}$ (except $\overline{\text{BUSY}}$)
	V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +3.5 \text{ mA}$ for D ₀ -D ₇
				0.4	V	$I_{OL} = +5 \text{ mA}$ for $\overline{\text{BUSY}}$
	I_{IL}	Input Leakage		±10	μA	$0.4 \leq V_{IN} \leq +2.4 \text{ V}$
	I_{OL}	Output Leakage		±10	μA	$0.4 \leq V_{OUT} \leq +2.4 \text{ V}$
				25	mA	Standby, AC = Static
	I_{CC}	V_{CC} Supply Current		40	mA	Fast Cycle Operation, TC = 400 ns

Capacitance	Capacitance of input or output pins, except $\overline{\text{BUSY}}$:	5 pF (max)
	Capacitance of $\overline{\text{BUSY}}$ input/output:	10 pF (max)
	All ac parameters assume a load capacitance of 100 pF maximum.	



Interfacing the Z6132 to a Z8000

The Z8001 or Z8002 CPU addresses memory as bytes, but can access either 8-bit bytes or 16-bit words. When writing a byte, A_0 selects the byte within a word; in all other cases the Z8000 always accesses a word, and A_0 is ignored. (When reading a byte, the memory reads a word and the CPU selects the appropriate byte internally.)

The odd- and even-byte memory banks use separate Chip Select decoders. The LS157 multiplexer is used as a function generator and

activates either the odd bank, the even bank or both, as determined by A_0 , Read/Write ($R/\bar{W}$) and Byte/Word ($B/\bar{W}$).

Address labels A_0 - A_{11} and Data labels D_0 - D_7 are used only to illustrate this example. Obviously, all address pins as well as all Data pins can be arranged arbitrarily to accommodate the PC board layout. A_0 must, however, always be connected to AD_1 to enhance the self-refresh operation.

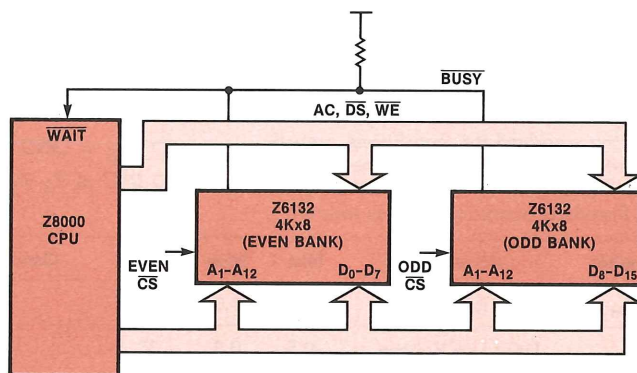


Figure 5. Block Diagram

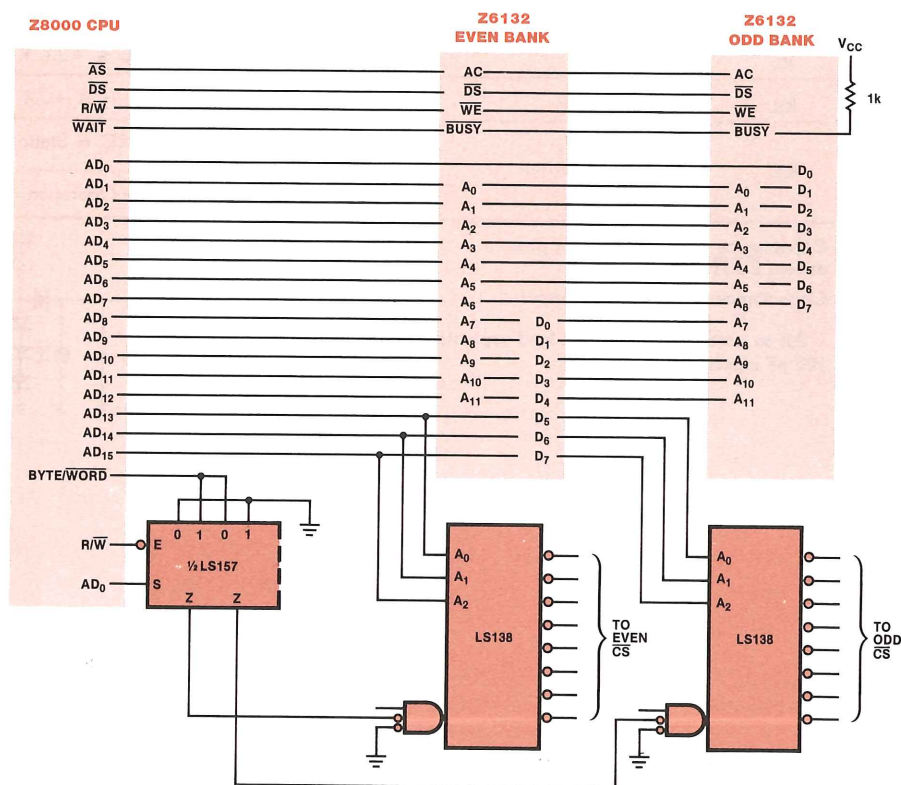


Figure 6. Connection Diagram

**Interfacing
the Z6132 to
a Z-80**

The $\overline{\text{MREQ}}$ output from the Z80 CPU is the main timing signal in this application. A D-type flip-flop is used to delay $\overline{\text{MREQ}}$, because the $\overline{\text{READ}}$ signal from the CPU is not guaranteed valid when $\overline{\text{MREQ}}$ goes Low. Synchronizing $\overline{\text{MREQ}}$ with the rising clock edge has another benefit: it lengthens the short $\overline{\text{MREQ}}$ signal during instruction fetch cycle by stretching it through the subsequent refresh cycle. (The refresh cycle of the Z80 is not used here, because the Z6132 provides its own refresh.)

AC is High for at least two clock periods, Low for at least one, thus Long Cycle Refresh Timing is possible even between the fastest Z-80A CPU timing (4 MHz) and the slowest 6132-5 memory.

$\overline{\text{DS}}$ is generated by inverting AC (an LS-TTL gate provides sufficient delay), and gating it with $\overline{\text{MREQ}}$ avoids bus contention. If the Z6132 is used for program storage, the critical timing parameter is the access time from Address Clock (parameter #13) during an M1 (opcode fetch) cycle.

**Interfacing
the Z6132 to
a Z8**

The Z6132 interfaces directly with the single chip Z8 microcomputer. Port 1 provides the 8-bit multiplexed Address/Data bus, and the more significant address bits are provided by Port 0.

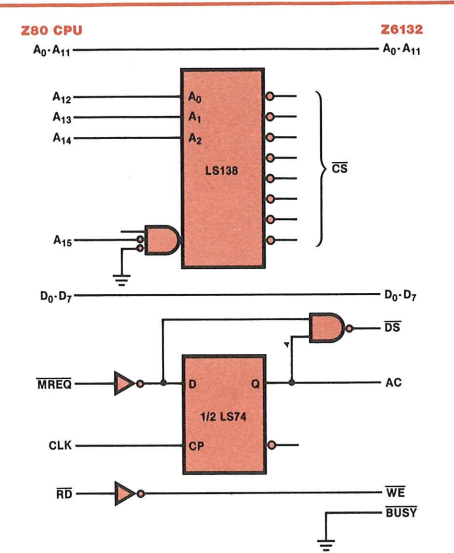


Figure 7. Z-80 Connection Diagram

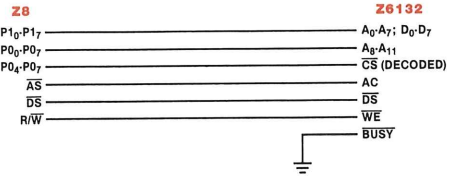


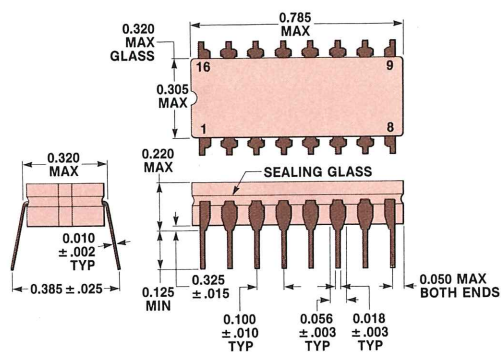
Figure 8. Z8 Connection Diagram

4

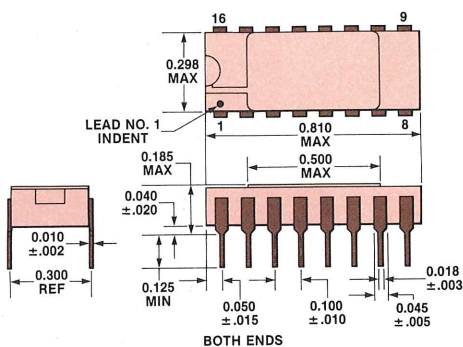
Package Dimensions

Zilog

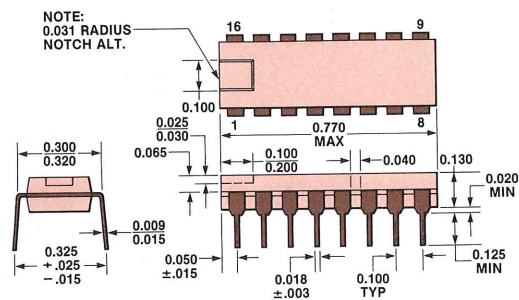
**Package
Dimensions**



16-Pin Cerdip Package

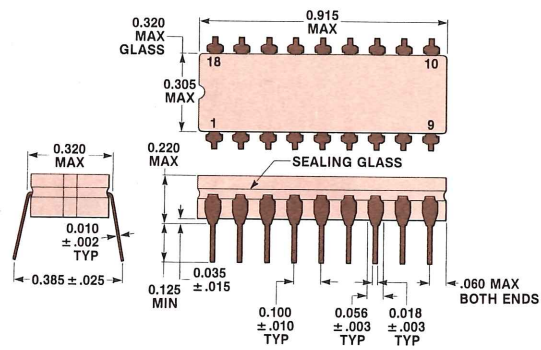


16-Pin Ceramic Package

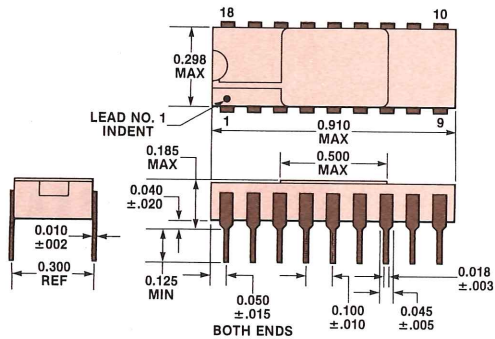


16-Pin Plastic Package

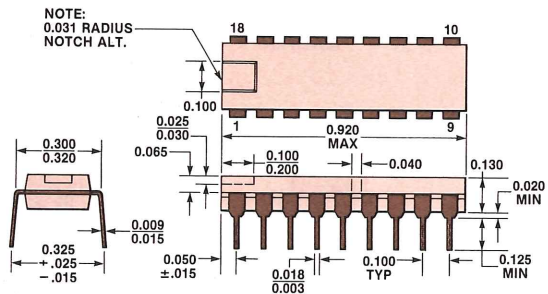
**Package
Dimensions**
(Continued)



18-Pin Cerdip Package

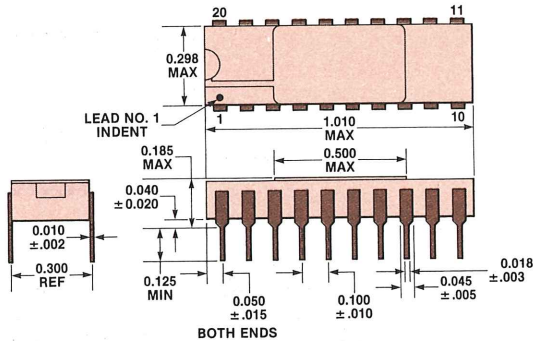


18-Pin Ceramic Package

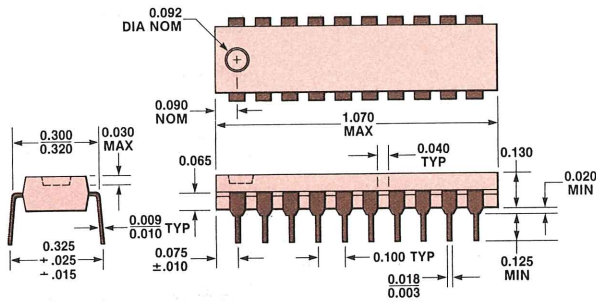


18-Pin Plastic Package

Package
Dimensions
(Continued)

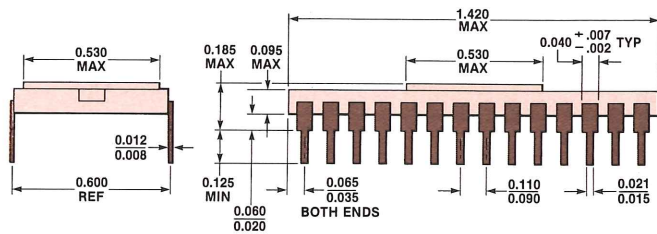
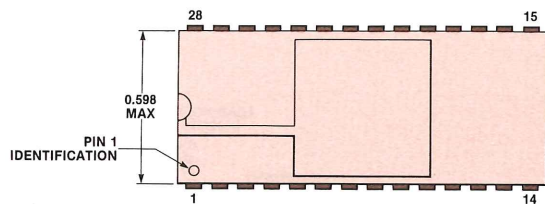


20-Pin Ceramic Package

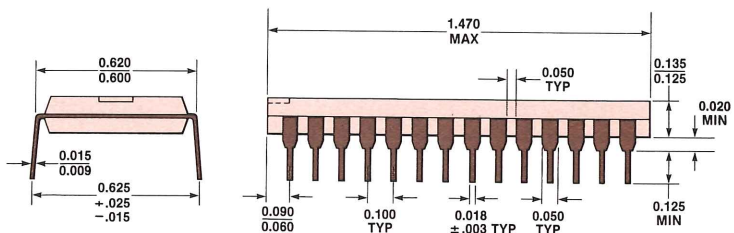
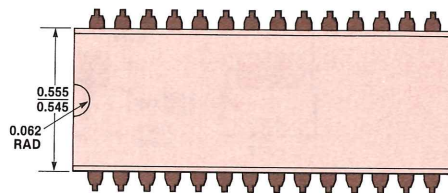


20-Pin Plastic Package

Package Dimensions (Continued)

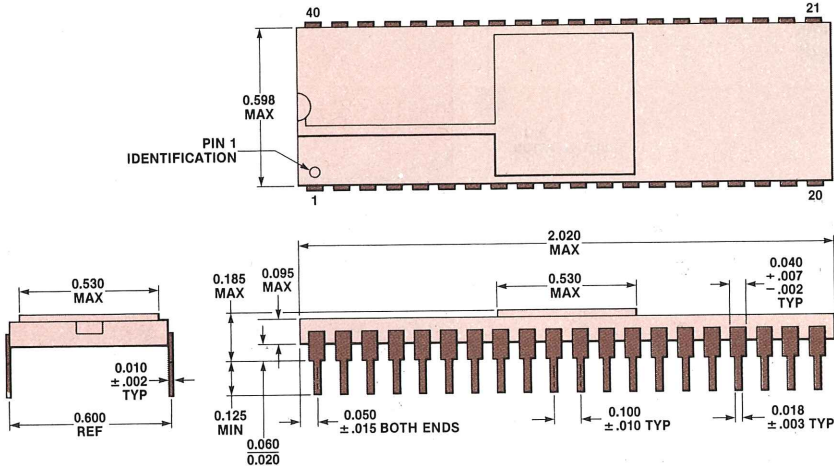


28-Pin Ceramic Package

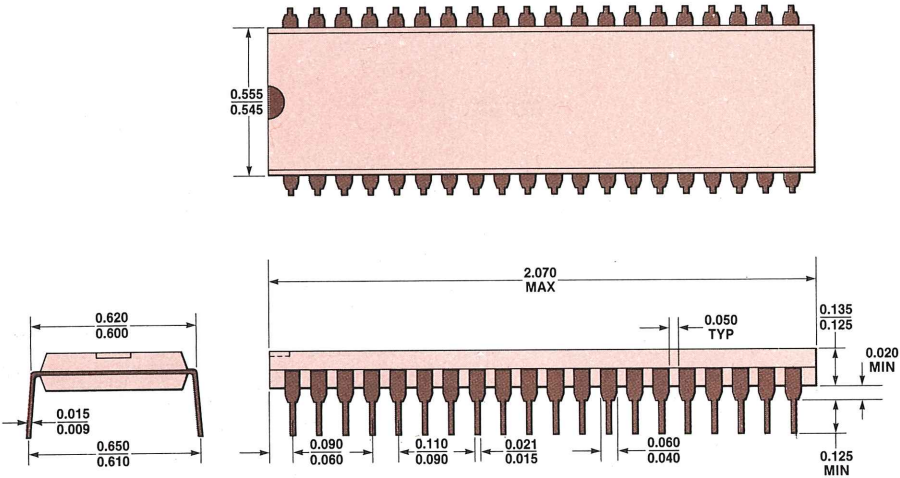


28-Pin Plastic Package

Package
Dimensions
(Continued)

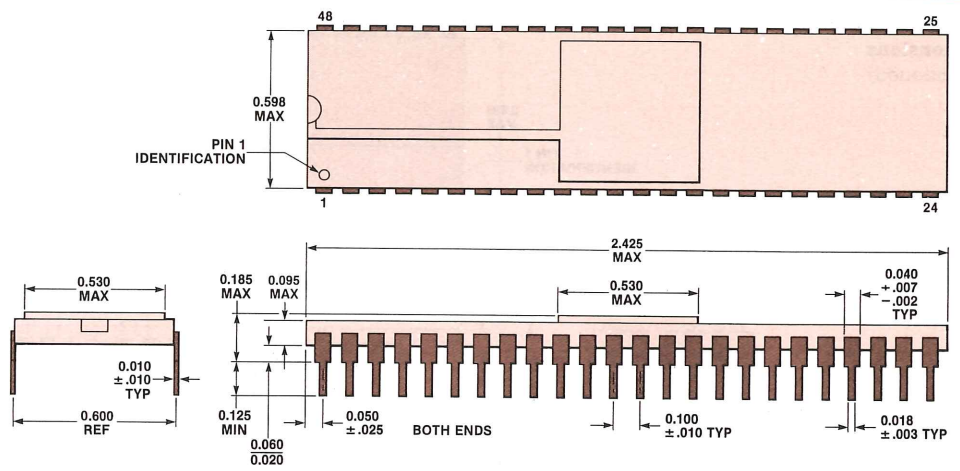


40-Pin Ceramic Package

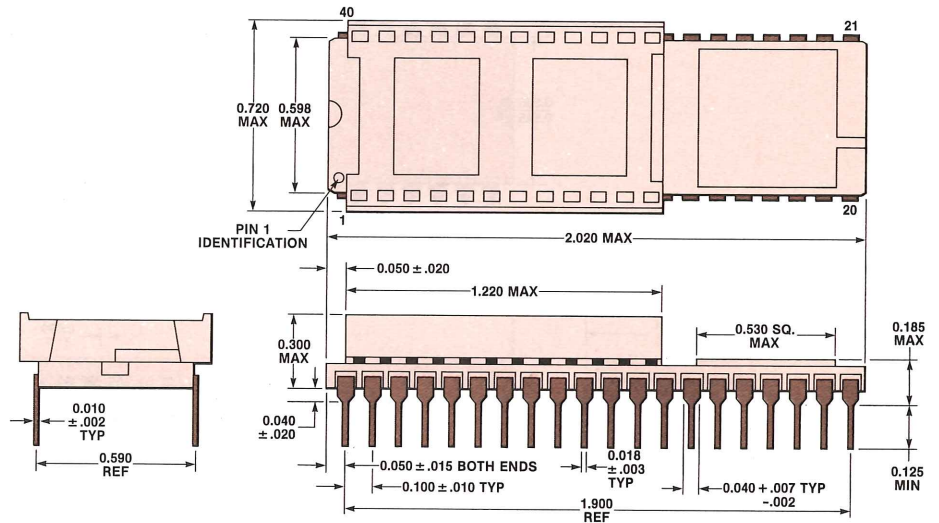


40-Pin Plastic Package

**Package
Dimensions**
(Continued)



48-Pin Ceramic Package



40-Pin Protopack Package

5

Sales Offices

Zilog

ZILOG SALES OFFICES

West

Sales & Technical Center
Zilog, Incorporated
1333 Lawrence Expressway
Santa Clara, CA 95051
Tele: (408) 446-9848

Sales & Technical Center
Zilog, Incorporated
Suite K
18023 Sky Park Circle
Irvine, CA 92714
Tele: (714) 549-2891
TWX: 910 595-2803

Sales & Technical Center
Zilog, Incorporated
6925 Canby Avenue
Suite 109
Reseda, CA 91335
Tele: (213) 708-3944
TWX: 910 493 2074

Midwest

Sales & Technical Center
Zilog, Incorporated
Suite 147
890 East Higgins Road
Schaumburg, IL 60195
Tele: (312) 885-8080
TWX: 910 291 1064

South

Sales & Technical Center
Zilog, Incorporated
2711 Valley View, Suite 103
Dallas, TX 75234
Tele: (214) 243-6550
TWX: 910 860 5850

Technical Center
Zilog, Incorporated
1442 U.S. Hwy 19 South
Suite 135
Clearwater, FL 33516

East

Sales & Technical Center
Zilog, Incorporated
Corporate Place
99 South Bedford St.
Burlington, MA 01803
Tele: (617) 273-4222
TWX: 710 332-1726

Sales & Technical Center
Zilog, Incorporated
110 Gibraltar Road
Horsham, PA 19044
Tele: (215) 441-8282
TWX: 510 665 7077

United Kingdom

Zilog (U.K.) Limited
Babbage House
Maidenhead SL6 1DU
Berkshire, United Kingdom
Tele: (0628) 36131
TELEX: 848609

West Germany

Zilog GmbH
Zugspitzstrasse 4
8011 Vaterstetten
Munich, West Germany
Tele: 08106 4035
TELEX: 529110 Zilog d.

Japan

Zilog, Japan KK
Linden Sky Heights
Bldg. 1F
13-2 Sakuragaoka-Machi
Shibuya-Ku Tokyo 150
Japan
Tele: 03-476-3010
TWX: 781 23723 Lawright

U.S. AND CANADA SALES REPRESENTATIVES

Legend:

A = All Products
B = Boards
C = Components
D = Development Systems
E = MCZ Systems

Headquarters

Zilog, Inc.
10340 Bubb Road
Cupertino, CA 95014
Phone (408) 446-4666
TWX 910-338-7621

Alabama

Electro Space, Inc.
303 Williams Rd.
Suite 122
Huntsville, AL 35801
Phone (205) 533-1735

Arizona

Thorson Desert States, Inc.
6560 N. Scottsdale Road
Suite E-204
Scottsdale, AZ 85253
Phone (602) 998-2444
TWX 910-950-1152

Arkansas

The Thorson Company (A)
4445 Alpha Road
Dallas, TX 75234
Phone (214) 233-5744
TWX 910-861-4423

California

Costar Inc. (B,C,D)
10080 N. Wolfe Road
Suite SW3-175
Cupertino, CA 95014
Phone (408) 446-9339
TWX 910-338-0206

Thorson West (M)
625 Ellis Street
Mountain View, CA 94043
Phone (415) 964-9300
TWX 910-379-6598

Littlefield and Smith Assoc. (A)
11772 Sorrento Valley Road
Suite 235
San Diego, CA 92121
Phone (714) 455-0055
TWX 910-322-1730

Interstate Marketing Assoc.
21048 Ventura Blvd.
Woodland Hills, CA 91365
Phone (213) 883-7606
TWX 910-494-1239

Colorado

Thorson Rocky Mountain (A)
7076 So. Alton Way, Bldg.D
Englewood, CO 80112
Phone (303) 779-0666
TWX 910-931-0429

Connecticut

John E. Boeing Co. (B,C,D)
18 Muzzey Street
Lexington, MA 02173
Phone (617) 862-2500
TWX 710-326-1774

(Fairfield County only)
ABC Electronic Sales Co. (A)
220 Marble Ave.
Pleasantville, NY 10570
Phone (914) 769-8088
TWX 510-222-4231

Delaware

Dyna Rep. Company (A)
6399 Little River Turnpike
Alexandria, VA 22312
Phone (703) 354-1222
TWX 710-832-0825

Florida

Dyne-A-Mark Corp. (A)
1001 N.W. 62nd Street
Suite 107
Fort Lauderdale, FL 33309
Phone (305) 771-6501
TWX 510-956-9872

Dyne-A-Mark Corp. (A)
405 S. Aurora
Clearwater, FL 33515
Phone (813) 441-4702
TWX 810-866-0438

Dyne-A-Mark Corp. (A)
303 East Sermoram Blvd.
Suite K
Alamonte Springs, FL 37201
Phone (305) 831-2097
TWX 810-853-5039

Georgia

Electro Space, Inc.
303 Williams Rd.
Suite 122
Huntsville, AL 35801
Phone (205) 533-1735

Idaho

Thorson Rocky Mountain (A)
7076 So. Alton Way, Bldg.D
Englewood, CO 80112
Phone (303) 779-0666
TWX 910-931-0429

Illinois

(Northern)
Contact Zilog's Central Office
Phone (312) 885-8080

(Southern)
Ensko-Rep Company (A)
7939 Floyd
Room L12
Overland Park, KS 66204
Phone (913) 381-7557
TWX 910-749-4066

Lancastle, LTD. (B,C,D)
111 North Avenue
Barrington, IL 60010
Phone (312) 381-6780
TWX 910-651-1900

Indiana

D.M. Assoc. (C,B,D)
4351 East 82nd Street
Suite D
Indianapolis, IN 46250
Phone (317) 842-4424
TWX 810-260-1438

Iowa

Ensko-Rep Company (A)
2615 W. Central Park
Davenport, IA 52804
Phone (319) 386-6600

Kansas

Ensko-Rep Company (A)
7939 Floyd
Room L-12
Overland Park, KS 66204
Phone (913) 381-7557
TWX 910-749-4066

Kentucky

D.M. Associates (C,B,D)
4351 East 82nd Street
Suite D
Indianapolis, IN 46250
Phone (317) 842-4424
TWX 810-260-1438

Louisiana

The Thorson Company (A)
4445 Alpha Road
Dallas, TX 75234
Phone (214) 233-5744
TWX 910-861-4423

Maine

John E. Boeing Co. (B,C,D)
18 Muzzey Street
Lexington, MA 02173
Phone (617) 862-2500
TWX 710-326-1774

Maryland

Dyna Rep Co. (A)
6399 Little River Turnpike
Alexandria, VA 22312
Phone (703) 354-1222
TWX 710-832-0825

Massachusetts

John E. Boeing Co. (B,C,D)
18 Muzzey Street
Lexington, MA 02173
Phone (617) 862-2500
TWX 710-326-1774

U.S. AND CANADA SALES REPRESENTATIVES (Continued)

Michigan

Greiner & Assoc. (C,D)
15324 E. Jefferson
Gross Point Pk., MI 48230
Phone (313) 499-0188

Minnesota

H.M.R. Inc. (A)
7200 France Ave. South
Minneapolis, MN 55435
Phone (612) 831-7400
TWX 910-576-2755

Mississippi

Electro Space, Inc.
303 Williams Rd.
Suite 122
Huntsville, AL 35801
Phone (205) 533-1735

Missouri

Ensco-Rep Company (A)
11714 Lackland Industrial Drive
St. Louis, MO 63141
Phone (314) 567-3935
TWX 910-764-0856

Montana

Thorson Rocky Mountain (A)
7076 So. Alton Way, Bldg.D
Englewood, CO 80112
Phone (303) 779-0666
TWX 910-931-0429

Nebraska

Ensco-Rep Company (A)
7939 Floyd
Overland Park, KS 66204
Phone (913) 381-7557
TWX 910-749-4066

Nevada

(Southern)
Thorson Desert States, Inc.
6560 North Scottsdale Road
Suite E-204
Scottsdale, AZ 85253
Phone (602) 998-2444
TWX 910-950-1152

(Northern)
Thorson West (A)
625 Ellis Street
Mountain View, CA 94043
Phone (415) 964-9300
TWX 910-379-6598

New Hampshire

John E. Boeing Co. (B,C,D)
18 Muzzey Street
Lexington, MA 02173
Phone (617) 862-2500
TWX 710-326-1774

New Jersey

ABC Electronic Sales Co. (A)
Route 94-RR, Box 397
Fredon 3X
Hamburg, NJ 07419
Phone (201) 827-9045
TWX 510-222-4231

(Southern)
Vantage Sales Co. (A)
21 Bala Avenue
Bala Cynwyd, PA 19004
Phone (215) 667-0990
TWX 510-662-5846

New Mexico

Trembly Associates (A)
119 Quincy N.E.
Albuquerque, NM 87108
Phone (505) 266-8616
TWX 910-989-1639

New York

(Metropolitan)
ABC Electronic Sales Co. (A)
99 Hillside Avenue
Williston Park, LI, NY 11596
Phone (516) 747-6610
TWX 510-222-4232

(Upstate)
L-Mar Associates Inc. (A)
315 Mt. Read Blvd.
P.O. Box 7945
Rochester, NY 14611/14606
Phone (716) 328-5240
TWX 510-253-0943

L-Mar Associates Inc. (A)
216 Tilden Drive
E. Syracuse, NY 13057
Phone (315) 437-7779

North Carolina

Over and Over Inc. (B,C,D)
500 Archdale Drive
Charlotte, NC 28210
Phone (704) 527-3070
TWX 810-621-0460

North Dakota

H.M.R. Inc. (A)
7200 France Avenue South
Minneapolis, MN 55435
Phone (612) 831-7400
TWX 910-576-2755

Ohio

Midwest Marketing Associates
35 Compark Rd.
Suite 201
Dayton, OH 45459
Phone (513) 433-2511
TWX 810-459-1610

Midwest Marketing Associates
26 N. Main St.
Chagrin Falls, OH 44022
Phone (216) 247-6655

Oklahoma

The Thorson Company (A)
4445 Alpha Road
Dallas, TX 75234
Phone (214) 233-5744
TWX 910-861-4423

Oregon

Quadra Sales Corp.
14825 N.E. 40th Street
Suite 340
Redmond, WA 98059
Phone (206) 883-3550
TWX 910-449-2592

Pennsylvania

(Eastern)
Vantage Sales Co. (A)
21 Bala Avenue
Bala Cynwyd, PA 19004
Phone (215) 667-0990
TWX 510-662-5846

Rhode Island

John E. Boeing Co. (B,C,D)
18 Muzzey Street
Lexington, MA 02173
Phone (617) 862-2500
TWX 710-326-1774

South Carolina

Over and Over Inc. (B,C,D)
500 Archdale Drive
Charlotte, NC 28210
Phone (704) 527-3070
TWX 810-621-0460

South Dakota

H.M.R. Inc. (A)
7200 France Avenue South
Minneapolis, MN 55435
Phone (612) 831-7400
TWX 910-576-2755

Tennessee

Electro Space, Inc.
303 Williams Rd.
Suite 122
Huntsville, AL 35801
Phone (205) 533-1735

Texas

The Thorson Company
Southwest (A)
6633 Hillcroft Avenue
Suite 105
Houston, TX 77081
Phone (713) 771-3504
TWX 910-881-3707

The Thorson Company
Southwest (A)
4445 Alpha Road
Dallas, TX 75234
Phone (214) 233-5744
TWX 910-861-4423

The Thorson Company
Southwest (A)
300 East Huntland Drive
Suite 103
Austin, TX 78752
Phone (512) 451-7527
TWX 910-874-1394

Utah

The Thorson Company (A)
3691 West 1987 South
Salt Lake City, UT 84104
Phone (801) 973-7969

Vermont

John E. Boeing Co. (B,C,D)
18 Muzzey Street
Lexington, MA 02173
Phone (617) 862-2500
TWX 710-326-1774

Virginia

Dyna Rep Co. (A)
6399 Little River Turnpike
Alexandria, VA 22312
Phone (703) 354-1222
TWX 710-832-0825

Washington

Quadra Sales Corp. (A)
14825 N.E. 40th St.
Suite 340
Redmond, WA 98052
Phone (206) 883-3550
TWX 910-449-2592

Wisconsin

Lancaster LTD.
6414 Copsps Avenue
Suite 148K
Madison, WI 53716
Phone (608) 222-7550

Wyoming

Thorson Rocky Mountain (A)
7076 So. Alton Way, Bldg.D
Englewood, CO 80112
Phone (303) 779-0666
TWX 910-931-0429

Canada

Munro Electronic Components
Ltd. (A)
7171 Torbram Road
Unit C9
Mississauga, Ontario L4T 3W4
Phone (416) 676-1042
TWX 610-492-8970

Munro Electronic Components
Ltd. (A)
4150 Blvd. St. Jean
Suite 307
Dollard des Ormeaux
Montreal, Quebec H9G 1X6
Phone (514) 626-0534

Munro Electronic Components
Ltd. (A)
R.R. 1
Dunrobin, Ontario K0A 1T0
Phone (613) 832-1640

National Electrolab Ltd.
1536 Columbia Street
North Vancouver, B.C. V7J 1A4
Phone (604) 985-0511
TLX 0435-2683 'Natalec'

U.S. AND CANADA SALES DISTRIBUTORS

Alabama

Hall-Mark Electronics
4739 Commercial Drive
Huntsville, AL 35805
Phone (205) 837-8700
TWX 810-726-2187

RM Electronics
4702 Governors Drive, S.W.
Huntsville, AL 35805
Phone (205) 852-1550
TWX 810-726-2177

Arizona

*Kierulff Electronics
4134 E. Wood Street
Phoenix, AZ 85040
Phone (602) 243-4101
TWX 910-951-1550

Sterling Electronics
P.O. Box 20867
UPS-2001 East University Drive
Phoenix, AZ 85036/85034
Phone (602) 258-4531
TLX 667-317

California

Anthem Electronics
2661 Dow Avenue
Tustin, CA 92680
Phone (714) 730-8000
Phone (213) 582-2122

Anthem Electronics
4040 Sorrento Valley Blvd.
San Diego, CA 92121
Phone (714) 279-5200
Phone (714) 453-9005
TWX 910-335-1515

Anthem Electronics
1020 Stewart Drive
Sunnyvale, CA 94086
Phone (408) 738-1111
TWX 910-339-9312

*Kierulff Electronics
2585 Commerce Way
Los Angeles, CA 90040
Phone (213) 725-0325
TWX 910-580-3106

*Kierulff Electronics
3969 E. Bayshore Rd.
Palo Alto, CA 94303
Phone (415) 968-6292
TWX 910-379-6430

Kierulff Electronics
8797 Balboa Avenue
San Diego, CA 92123
Phone (714) 278-2112
TWX 910-335-1182

Kierulff Electronics
14101 Franklin Avenue
Tustin, CA 92680
Phone (714) 731-5711
TWX 910-595-2599

*Western Microtechnology
10040 Bubb Road
Cupertino, CA 95014
Phone (408) 725-1660

Colorado

Kierulff Electronics
10890 E. 47th Avenue
Denver, CO 80239
Phone (303) 371-6500
TWX 910-932-0169

Bell Industries
8155 West 48th Avenue
Wheatridge, CO 80033
Phone (303) 424-1985
TWX 910-938-0393

R.V. Weatherford Company
8515 E. Orchard Road
Englewood, CO 80111
Phone (303) 770-9762
TWX 910-933-0173

Connecticut

Wilshire Electronics
Barnes Industrial Park
Village Lane
Wallingford, CT 06492
Phone (203) 265-3822
TWX 710-476-0300

Florida

Hall-Mark Electronics
1302 West McNab Road
Fort Lauderdale, FL 33309
Phone (305) 971-9280
TWX 510-956-9720

Hall-Mark Electronics
7233 Lake Ellenor Drive
Orlando, FL 32809
Phone (305) 855-4020
TWX 810-850-0183

Kierulff Electronics, Inc.
3247 Tech Drive
St. Petersburg, FL 33702
Phone (813) 576-1966
TWX 810-863-5625

Georgia

*Wilshire Electronics
4364-B Schackelford Road
Norcross, GA 30093
Phone (404) 923-5750

Illinois

Hall-Mark Electronics
1177 Industrial Drive
Bensonville, IL 60106
Phone (312) 860-3800
TWX 910-222-3645

*Kierulff Electronics
1536 Landmier Road
Elk Grove Village, IL 60007
Phone (312) 640-0200
TWX 910-222-0351

RM Electronics
265 Eisenhower Lane South
Lombard, IL 60148
Phone (312) 932-5150

Indiana

RM Electronics
5545-K West Raymond
Indianapolis, IN 46241
Phone (317) 247-9701

Kansas

Hall-Mark Electronics
11870 West 91st Street
Congleton Industrial Park
Shawnee Mission, KS 66214
Phone (913) 888-4747
TWX 910-749-6620

Maryland

Hall-Mark Electronics
6655 Amberton Drive
Baltimore, MD 21227
Phone (301) 796-9300
TWX 710-862-1942

Wilshire Electronics
1037 Taft Street
Rockville, MD 20850
Phone (301) 340-7900

Massachusetts

*Kierulff Electronics
13 Fortune Drive
Billerica, MA 01821
Phone (617) 667-8331
TWX 710-390-1449

*Wilshire Electronics
One Wilshire Road
Burlington, MA 01803
Phone (617) 272-8200
TWX 710-332-6359

Michigan

RM Electronics
4310 Roger B. Chaffee
Memorial Dr.
Grand Rapids, MI 49508
Phone (616) 531-9300
TWX 810-273-8779

Minnesota

Hall-Mark Electronics
9201 Penn Avenue South
Suite 10
Bloomington, MN 55435
Phone (612) 884-9056
TWX 910-576-3187

Kierulff Electronics
5280 West 74th
Edina, MN 55435
Phone (612) 835-4388
TWX 910-576-2721

Missouri

Hall-Mark Electronics
13789 Rider Trail
Earth City, MO 63045
Phone (314) 291-5350
TWX 910-762-0672

New Jersey

Kierulff Electronics
3 Edison Place
Fairfield, NJ 07006
Phone (201) 575-6750
TWX 710-734-4372

Lionex
23 Just Road
Fairfield, NJ 07006
Phone (201) 227-7960
TWX 710-734-4312

*Wilshire Electronics
1111 Paulison Avenue
Clifton, NJ 07015
Phone (201) 340-1900
TWX 710-989-7052

Wilshire Electronics
102 Gaither Drive
Mt. Laurel, NJ 08054
Phone PA (215) 627-1920
Phone NJ (609) 234-9100

New Mexico

Bell Industries
11728 Linn N.E.
Albuquerque, NM 87123
Phone (505) 292-2700
TWX 910-989-0625

Sterling Electronics
3540-D Pan American Freeway
N.E.
Albuquerque, NM 87107
Phone (505) 345-6601
TLX 66-04-41

New York

(Metropolitan)
*Wilshire Electronics
110 Parkway South
Hauppauge, LI, NY 11787
Phone (516) 543-5599
TWX 510-226-3782

Lionex Corporation
415 Crossways Park Drive
Woodbury, LI, NY 11798
Phone (516) 921-4414

Summit
916 Main Street
Buffalo, NY 14202
Phone (716) 884-3450
TWX 110-522-1692

Wilshire Electronics
10 Hooper Road
Endwell, NY 13760
Phone (607) 754-1570
TWX 510-252-0194

Wilshire Electronics
1260 Scottsville Road
Rochester, NY 14624
Phone (716) 235-7620
TWX 510-253-5526

North Carolina

Hall-Mark Electronics
1208 Front Street
Building K
Raleigh, NC 27609
Phone (919) 832-4465
TWX 510-928-1831

Ohio

Hall-Mark Electronics
6969 Worthington Galena Road
Worthington, OH 43085
Phone (614) 846-1882

Hall-Mark Electronics Corp.
370 Alpha Park
Highland Heights, OH 44143
Phone (216) 473-2907

CAM/Ohio
749 Miner Rd.
Cleveland, OH 44143
Phone (216) 461-4700
TWX 810-427-2976

Oklahoma

Hall-Mark Electronics
5460 South 103rd East Avenue
Tulsa, OK 74145
Phone (918) 835-8458
TWX 910-845-2290

Oregon

Kierulff Electronics
14273 N.W. Science Park Drive
Portland, OR 97229
Phone (503) 641-9150

U.S. AND CANADA SALES DISTRIBUTORS (Continued)

Pennsylvania

Hall-Mark Electronics
Pike Industrial Park
Huntington Valley, PA 19006
Phone (215) 355-7300
TWX 510-667-1750

CAM/RPC
RIDC Park 620 Alpha Dr.
Pittsburgh, PA 15238
Phone (412) 782-3770
TWX 710-795-3126

CAM/RPC Computer Systems Co.
RIDC Park
620 Alpha Dr.
Pittsburgh, PA 15238
Phone (412) 782-3770
TWX 710-795-3126

Texas

Hall-Mark Electronics
10109 McKalla Place
Suite F
Austin, TX 78758
Phone (512) 837-2814
TWX 910-874-2031

RM Dallas Inc.
13534 Method Street
Dallas, TX 75243
Phone (214) 263-8361
Phone (214) 253-3100
TWX 910-867-4720

Hall-Mark Electronics
11333 Pagemill Drive
P.O. Box 222035
Dallas, TX 75222
Phone (214) 234-7300

Hall-Mark Electronics
8000 Westglen
Houston, TX 77063
Phone (713) 781-6100
TWX 910-881-2711

Utah

Bell Industries
3639 West 2150 South
Salt Lake City, UT 84120
Phone (801) 972-6969
TWX 910-925-5686

Kierulff Electronics
3695 West 1987 South
Salt Lake City, UT 84104
Phone (801) 973-6913

Washington

Kierulff Electronics
1005 Andover Park East
Tukwila, WA 98188
Phone (206) 575-4420
TWX 910-444-2034

*Sterling Electronics
1046 Industry Drive
Seattle, WA 98188
Phone (206) 575-1910
TLX 32-9652

Wisconsin

Hall-Mark Electronics
9657 South 20th St.
Oak Creek, WI 53154
Phone (414) 761-3000
TWX 910-262-3186

Kierulff Electronics
2212 E. Moreland Blvd.
Waukesha, WI 53186
Phone (414) 784-8160

Canada

*Future Electronics
5647 Ferrier Street
Montreal, Quebec H4P 2K5
Phone (514) 735-5775
TWX 610-412-3251

Future Electronics
4800 Dufferin Street
Downsview, Ontario M3H 5S8
Phone (416) 663-5563

Future Electronics
Baxter Center
1050 Baxter Road
Ottawa, Ontario K2C 3P2
Phone (613) 820-9471

Future Electronics
3070 Kings Way
Vancouver, B.C., CN V5R 5J7
Phone (604) 438-5545
TWX 610-922-1668

*demonstration center

ZILOG INTERNATIONAL DISTRIBUTORS

Australia

ZAP Systems Pty., Ltd.
51-53 Chandos Street
St. Leonards, N.S.W. 2065
Telephone: (2) 438-4533
Telex: (790) 27031 ZAP

Hong Kong

Saturn/Caduceus Industries, Ltd.
Hang Lung Centre
Room 1410
2-20 Paterson Street
Causeway Bay, Hong Kong
Telephone: (5) 77-0504
Telex: (780) 83640 CADUC HX

India

Kryonix
Kowdiar, Trvandrump 695003
South India
Telephone: 63805
Telex: (953) 884307 KRVIN

SRI Ram Associates
Los Fuentos Office Center
2343 Homestead Road
Bldg. D, Suite 3
Santa Clara, CA 95050
USA
Telephone: (408) 738-2295
Telex: 348369

Japan

Kanematsu Electronic
Components Corp.
Takanawa-Kikari Bldg.
19-26 Takanawa 3-Chome
Minato-Ku, Tokyo 108
Japan

Telephone: (3) 447-4141
Telex: (Japan) (781) 22334
TWX: (San Francisco)
910 372 6078

Matsushita Electric
Trading Co., Ltd.
32nd Floor,
World Trade Center Bldg.
No. 4-1, Hamamatsu Cho,
2-Chome
Minato-Ku, Tokyo 105
Japan

Telephone: (3) 435-4580
Telex: (781) 24841 MATSUELE

Nihon Teksel Co., Ltd.
Kyoshin Bldg.
13-14 Sakuragaoka-Machi
Shibuya-Ku Tokyo 150
Japan
Telephone: (3) 461-5121
Telex: (781) 23723 LAWRIGHT

Korea

Duksung Trading Co.
Rm 508 Sindo, Bldg.
65-4 2KA Chungmu-RO
Chung-KU Seoul, Korea
Telephone: (2) 778-3198
Telex: 23459 DUKSUNG

Latin America

Intectra
2349 Charleston Road
Mountain View, CA 94043
USA
Telephone: (415) 967-8818
Telex: 345545 INTECTRA MNTV

Malaysia

Unidata SDM BHD
6th Fl. Syed Kechik
Foundation Bldg.
Bangar, Kuala Lumpur 22-16
Malaysia
Telephone: 940666
Telex: MA31252

Singapore, Malaysia, Indonesia

UNIDATA PTE, Ltd.
Suite 12-201-324, 12th Fl.
Merlin Plaza, Beach Road
Singapore 0719
Telephone: (65) 2953281
Telex: (786) 25486

South Africa

Tecnetics Pty., Ltd.
P.O. Box 56310
Pinetown 2123
Randburg 2001
Transvaal, South Africa
Telephone: (11) 789-1121
Telex: (964) 89708 SA

Taiwan

Multitech American Company
19046 Bonnet Way
Saratoga, CA 95070
USA
Telephone: (408) 867-6201

Multitech International Corp.
2nd Floor
977 Min Shen E. Road
Taipei 105, Taiwan
Republic of China
Telephone: (2) 768-1232
Telex: (785) 23756 MULTIC

ZILOG EUROPEAN DISTRIBUTORS

A2M (005)
18 Avenue Dutartre
78150 Le Chesnay
France

Ditz Schweitzer A.S. (038)
Vallenbrackvej 41,
DK 2600
Glastrup
Denmark

Finn Metric OY (004)
Box 35
SF-02101 ESP00 10
Finland

Haliagraph p.v.b.a. (002)
Guldensporenlaan 19
B-9220 Merelbeke
Belgium

Jermyn GmbH (033)
Postfach 1180
D-6277 Camberg
West Germany

Jermyn Mogul Dist. (039)
Vestry Road
Sevenoaks
Kent
United Kingdom

Kontron Austria (001)
Industriestrasse B13
A2345 Brunn AM Gebirge
Austria

Kontron Elektronik GmbH (015)
Breslauer Str. 2
D-8057 Eching Bei Munich
West Germany

Memec Ltd. (017)
Thame Park Industrial Estate
Thame, Oxon
OX9 3RS
United Kingdom

Metric A/S (008)
Ekebergveien 312
Oslo 11, Norway

Microelectro Developments
International Ltd.
87-93 Wellington St.
Luton, Beds.
United Kingdom

Micropower Ltd. (013)
Hampstead House
Basingstoke, Hants RG21 1LG
United Kingdom

Microscan GmbH (029)
Ueterseering 31
2000 Hamburg 60
West Germany

Microsystems
64, Dimofontos St.
Athens 311, Greece

Scandia Metric AB (010)
Banvaktsvagen 20
Fack 171 19
Solna 1, Sweden

S.C. Metric A/S (003)
Skodsborgvej 350
DK-2850 Naerum
Denmark

SGS-ATES Componenti
Electronici S.P.A. (036)
via C. Olivetti 2
20041 Agrate Brianza
Milano
Italy

SGS-ATES Deutschland
Halbleiter
Bauelemente GmbH
Haidling, 17 8018 Grafing
Munich, West Germany

SGS-ATES France S.A. (035)
75643 Paris Cedex 13
"Le Platino"
17 Avenue De Choise
France

SGS-ATES (United Kingdom)
Ltd. (0)
Walton Street
Aylesbury, Bucks

SGS-ATES Scandinavia AB (034)
Tingwallavagen 9J-Box 30
19501 Maersta
Sweden

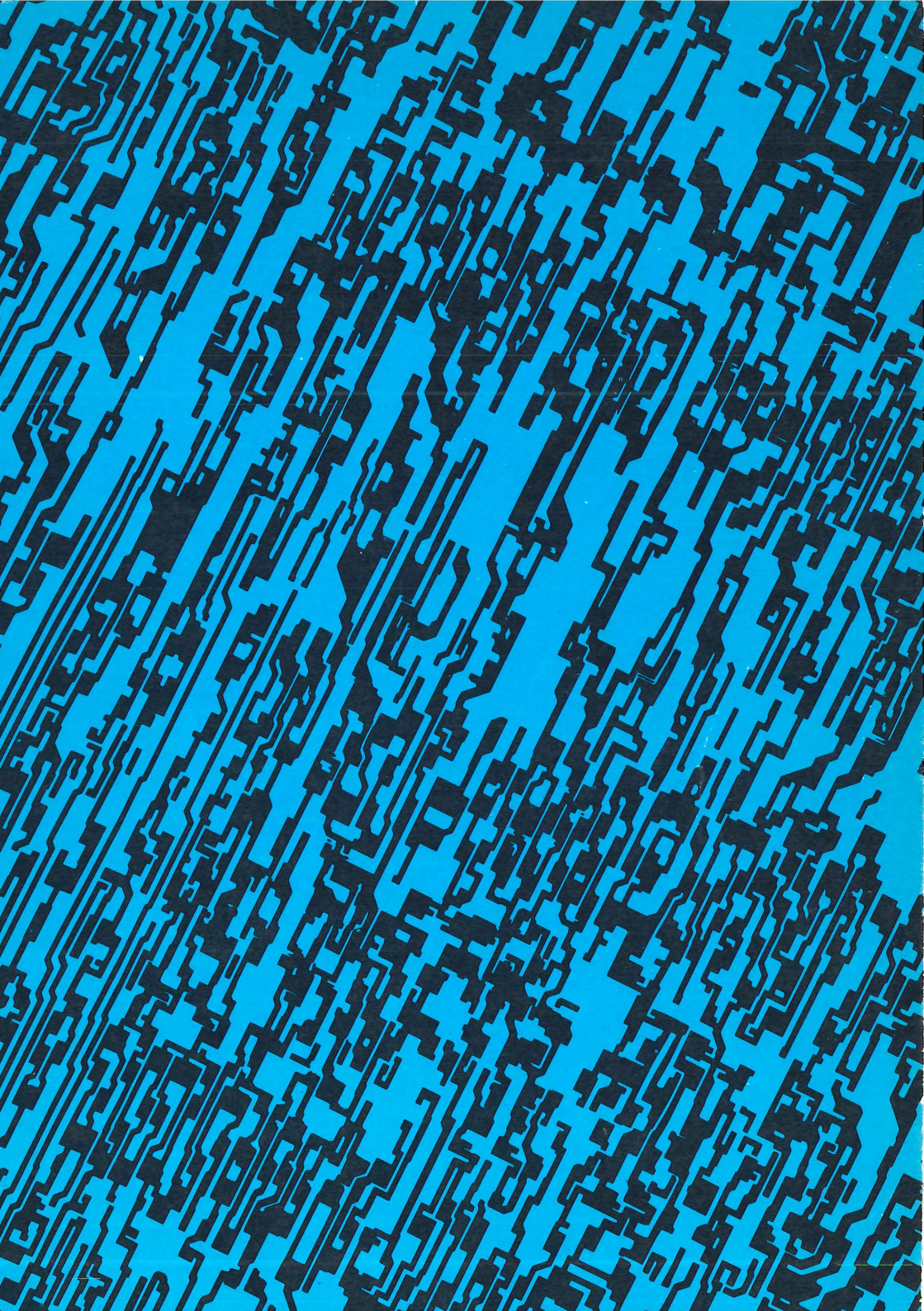
W. Stolz AG (011)
Bellikonstrasse 218
CH-8967 Widen-Mutschellen
Switzerland

Tekelec Airtronic B.V. (007)
P.O. Box 63
2700 AB Zcetermeer
The Netherlands

Unitronics S.A. (009)
Torre de Madrid, PL. 12, Ofc. 9
Madrid, Spain

Vectronics Ltd. (032)
60 Gordon St.
P.O. Box 16335
Tel-Aviv, Israel

Zelco SRI (028)
via V Monti 21
20123
Milan, Italy



Zilog, Inc.

10460 Bubb Road

Cupertino, California 95014

Telephone (408) 446-4666

TWX 910-338-7621